JFlex

JFlex

O JFlex é um gerador de analisadores léxicos escrito em Java, baseado no flex para C.

O JFlex constrói o analisador léxico a partir de um arquivo de especificação.

O JFlex gera uma classe em Java que faz a análise léxica.

JFlex

Como escrever uma especificação que será processada pelo JFlex ?

A especificação é dividida em 3 partes que são separadas por “%%”

Código do usuário
%%
Opções do JFlex
Declarações de Macros
%%
Regras léxicas

JFlex

Como escrever uma especificação que será processada pelo JFlex ?

A primeira parte denominada “Código do usuário” pode ser usada, por exemplo, para:

- Definir o pacote ao qual o lexer vai pertencer.
- Definir imports.
- Comentários de documentação.

Exemplo de código do usuário:

```
package lexer;
import java.util.Vector;
```

JFlex

Como escrever uma especificação que será processada pelo JFlex ?

A segunda parte, situada entre a primeira ocorrência de %% e a segunda, é usada para diretivas e macros.

Pode conter opções do JFlex ou macros de expressões regulares.

...
%%

%%
...

JFlex

Algumas opções que podem ser colocadas na segunda parte da especificação:

%class <name> - define o nome da classe do analisador léxico que vai ser gerada. Se não for definido, a classe se chamará yylex.

%{ e }% - delimitam código Java a ser inserido dentro da classe do analisador léxico.
Ex: atributos e métodos adicionais.

%cup - torna o arquivo gerado compatível com o gerador de parsers CUP.

%function <name> - define o nome da função que retorna o próximo token. Se não for definida, será yylex().

%type <class name> - define qual classe (já existente) irá representar os tokens.

%line - ativa o contador de linhas (atributo yyline).

%column - ativa o contador de colunas (atributo yycolumn).

Existem também opções para: tratamento de exceção, criação de estados léxicos, etc.

JFlex

Algumas opções que podem ser colocadas na segunda parte da especificação:

Macros

As macros definem nomes para expressões regulares particulares.

```
...
%%
ALPHA = [A-Za-z]
DIGIT = [0-9]
%%
...
```

As macros são definições auxiliares, mas não representam tokens. Podem ser usadas na definição dos tokens na terceira parte da especificação.

Para usar a definição de uma macro, deve-se delimitar seu nome por chaves.
Exemplo:

```
ALPHA_NUMERIC = {ALPHA}|{DIGIT}
```

JFlex

Algumas opções que podem ser colocadas na terceira parte da especificação:

- As regras estão representadas na terceira parte da especificação.

- Seção após o segundo “%%” que especifica regras léxicas no formato abaixo:

<expressão regular> { <código Java> }
...
%%
...
%%

[0-9] {System.out.println(“Digito encontrado”);}

- As expressões regulares podem usar as macros definidas na parte 2.

- A ação associada à expressão regular pode ser qualquer código Java.

JFlex

Algumas opções que podem ser colocadas na terceira parte da especificação:

Para expressões que representam tokens, a regra deve ter um “return” passando o token.

Se for uma expressão que representa comentários ou espaços brancos, a ação não deve retornar valor.

Se for preciso recuperar o lexema (um objeto String), basta chamar a função yytext().

Exemplo:

```
...
%%

"+"       {  return new Symbol(sym.MAIS);  }
[0-9]+    {  return new Symbol(sym.NUMERO, yytext());  }
[ \n\t\r] {  }
. {  }
```

JFlex

Algumas opções que podem ser colocadas na terceira parte da especificação:

Como criar intervalos de identificação de tokens na especificação das regras:

[0-9]*
ocorrência de 0 ou mais números

[0-9]+
ocorrência de 1 ou mais números

[0-9]?
ocorrência de 0 ou 1 número

[0-9]{1,4}
ocorrência de 1 a 4 números

JFlex

Algumas expressões regulares que podem ser colocadas na terceira parte da especificação:

Expressão	Significado
a	Caractere ‘a’
“foo”	Cadeia “foo”
[abc]	‘a’, ‘b’ ou ‘c’
[a-d]	‘a’, ‘b’, ‘c’ ou ‘d’
[^ab]	Qualquer caractere exceto ‘a’ e ‘b’
.	Qualquer caractere **exceto \n**
x | y	Expressão x ou y
xy	Concatenação
x*	Fecho de Kleene
x+	Fecho positivo
x?	Opcional
!x	**Negação**
~x	**Tudo até x (inclusive)**

JFlex

Algumas expressões regulares que podem ser colocadas na terceira parte da especificação:

^	→	caracter que o segue inicia o elemento
$	→	caracter que o antecede finaliza o elemento
.	→	caracter é um símbolo qualquer (exceto nova linha)
|	→	enumerador de alternativas
( )	→	agrupador
[ ]	→	especificificador de classes
*	→	caracter ocorre 0 ou mais vezes
+	→	caracter ocorre 1 ou mais vezes
?	→	caracter ocorre 1 ou 0 vezes
{n}	→	caracter ocorre exatamente "n" vezes
{n,}	→	caracter ocorre pelo menos "n" vezes
{n,m}	→	caracter ocorre pelo menos "n" vezes e não mais que "m" vezes

JFlex

Como utilizar o JFlex ?

Site oficial do JFlex:
http://jflex.de

Instruções de instalação do JFlex:
http://jflex.de/installing.html

Manual do JFlex:
https://jflex.de/manual.html

Obs.: é necessário ter o Java instalado na sua máquina.

JFlex

Como utilizar o JFlex na linha de comando:

1) Gerar o analisador léxico.
jflex Exemplo1.flex
(o resultado deste comando será a criação do arquivo Exemplo1.java, que contém o analisador léxico em java)

2) Compilar o analisador léxico que está no arquivo Exemplo1.java.
javac Exemplo1.java

3) Criar um arquivo texto com o código fonte a ser analisado pelo analisador léxico.
codigo_fonte_exemplo1.txt

4) Executar o analisador léxico (Exemplo1.class) para analisar o arquivo do código fonte (codigo_fonte_exemplo1.txt).
java Exemplo1 codigo_fonte_exemplo1.txt