UNIVERSIDADE DO OESTE DE SANTA CATARINA

CAMPUS DE SÃO MIGUEL DO OESTE

CURSO: CIÊNCIAS DA COMPUTAÇÃO

DISCIPLINA: COMPILADORES

PROFESSOR: JOHNI DOUGLAS MARANGON

Pratica JFlex

JFlex é uma ferramenta que permite gerar um analisador léxico a partir da *linguagem Lex.* A saída é uma classe Java.

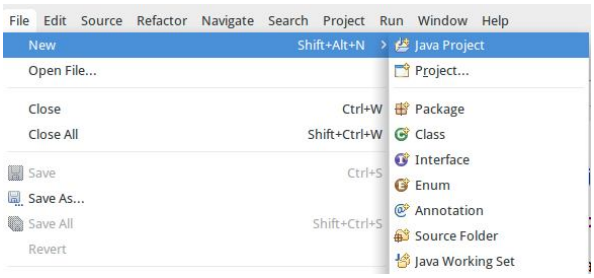
Faça o download das bibliotecas do JFlex nesse link: http://jflex.de/download.html

Nesse exemplo vamos utilizar a ferramenta integrada com a IDE do Eclipse.

Prática – criando o primeiro analisador léxico

1. Criando um novo projeto.

   No IDE do Eclipse crie um novo projeto (File -> New -> Java Project).

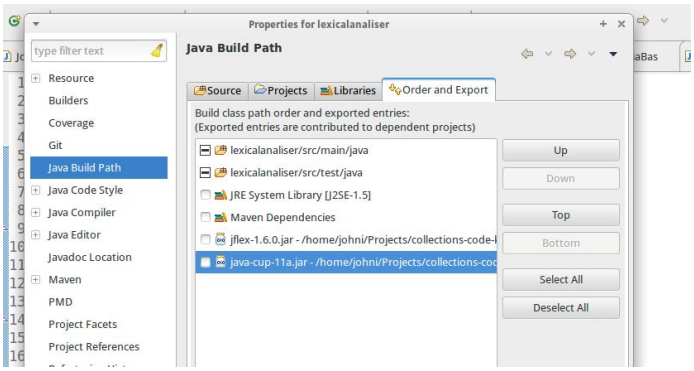


   Nome do projeto - **AnalisadorLexico**

2. Importe os arquivos **jar** do JFlex para o seu projeto.

   Baixar o JFlex 1.6.0 http://jflex.de/download.html e descompactar o arquivo em no C:\.

   Adicionar ao Build Path as bibliotecas “jflex-1.6.0.jar” e “java-cup-11a.jar”.



3. Criando a classe que irá gerar no analisador léxico.

Após o projeto criado crie uma classe chamada **Gerador**. Essa classe deve ter o mento **main** implementado. Defina o nome do pacote como **br.edu.unoesc.**

Digite o seguinte código nessa classe.

```
import java.io.File;

public class Gerador
{
    public static void main( String[] args )
    {
        String path = "c://";
        String arquivo = path + "linguagem.lex";

        File file = new File(arquivo );
        jflex.Main.generate(file);

    }
}
```

Essa classe será responsável por gerar o código que implementa as regras de linguagem de programação.

4. Criando o arquivo Lex.

O arquivo Lex possui as definições da nossa linguagem de programação.

Para criar esse arquivo na IDE do Eclipse vá em File -> New -> Other -> File. De o nome de **linguagem.flex**.

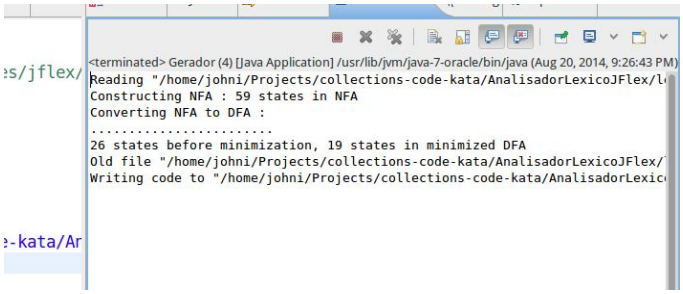
No arquivo digite o seguinte conteúdo.

```
package br.edu.unoesc;
import java_cup.runtime.*;

%%
```

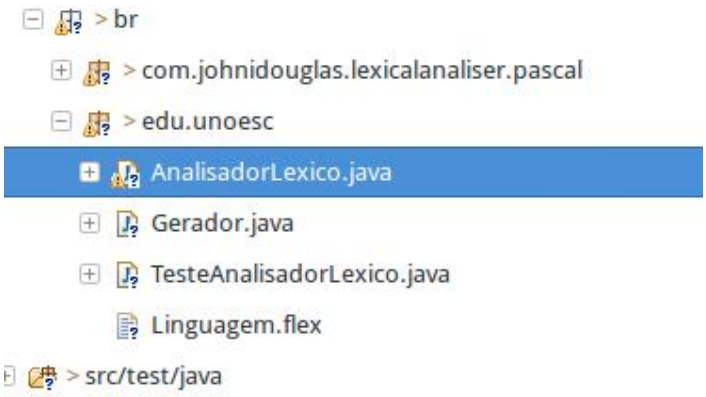
```
%{

/*-*
 * funcoes e variaveis
 */

private void imprimir(String descricao, String lexema) {
        System.out.println(lexema + " - " + descricao);

}

%}

/*-*
 * informacoes sobre a clase gerada
 */

%public
%class AnalisadorLexico
%type void


/*-*
 * definicaoes de regulares
 */
BRANCO = [\n| |\t]
ID = [_|a-z|A-Z][a-z|A-Z|0-9|_]*
INTEIRO = 0|[1-9][0-9]*
PONTOFLUTUANTE = [0-9][0-9]*"."[0-9]+
OPERADORES_MATEMATICOS = ("+" | "-" | "*" | "/")

%%

"if"                      { imprimir("Intrucao if", yytext()); }
"then"                    { imprimir("Intrucao then", yytext()); }
{BRANCO}                  { imprimir("Branco", yytext()); }
{ID}                      { imprimir("Identificador", yytext()); }
{INTEIRO}                 { imprimir("Numero", yytext()); }
{PONTOFLUTUANTE}          { imprimir("Ponto plututante", yytext()); }
{OPERADORES_MATEMATICOS}  { imprimir("Operadores matematatico", yytext()); }

"=="                      { imprimir("Operador igualdade", yytext()); }

. { throw new RuntimeException("Caractere invalido \""+yytext() +
                                        "\" na linha "+yyline+", coluna "+yycolumn); }
```

5. Testando o analisador criado.

Para ter o analisador é necessário executar a classe **Gerador** que possui o método **main**. Ao executar essa classe um resultado parecido com o da imagem abaixo será gerado.



Observe que no seu projeto foi criado a classe **AnalisadorLexico.** Essa classe possui a implementação das regras do arquivo Lex.

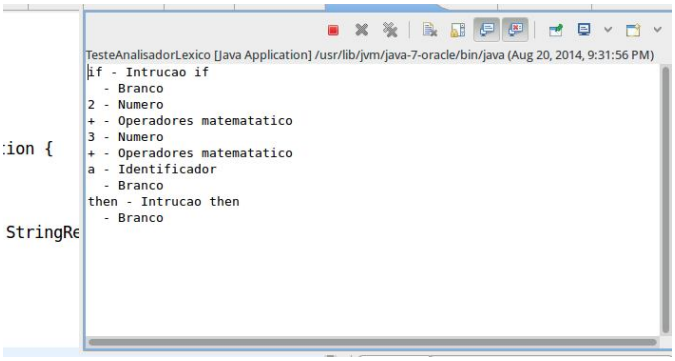


Agora vamos criar a classe **TesteAnalisadorLexico** que irá implementar de fato o nosso analisador léxico. File -> New -> Class.

A classe deve implementar o seguinte código

```
public class TesteAnalisadorLexico {

    public static void main( String[] args ) throws IOException {

        String expr = "if 2+3+a then ";

        AnalisadorLexico lexico = new AnalisadorLexico (new StringReader(expr));
        lexico.yylex();
    }
}
```

Execute essa classe o observe que a saída será a seguinte.



Prática – criando um analisador léxico para reconhecer um programa em Pascal

1. No projeto **AnalisadorLexico** crie uma classe chamada **GeradorPascal**. Essa classe deve ter o método **main** implementado e deve estar no pacote **br.edu.unoesc.pascal**. Semelhante a classe **Gerador** do exercício anterior.

```java
import java.io.File;

public class GeradorPascal {
    public static void main(String[] args) {

        String path = "c://";
        String arquivo = path + "pascal/Pascal.lex";

        File file = new File(arquivo);
        jflex.Main.generate(file);

    }
}
```

2. Crie o arquivo de código **programa.pas**.

```
program
  var Idade : integer;
  var ValorSalario: real;
begin
   if (Idade > 10) then
      ValorSalario := 100;
   else
      ValorSalario := ValorSalario * 2;
end.
```

3. Criar a classe **PascalToken** essa classe deve ter o seguinte código. Essa classe é auxiliar e será utilizada para representar cada token.

```java
public class PascalToken {

    public String valor;
    public String nome;

    public PascalToken(String nome, String valor) {
       this.valor = valor;
       this.nome = nome;
    }
}
```

4. Criar a classe **PascalAnalisador** com o seguinte código

```java
public class PascalAnalisador {

    public static void main(String[] args ) {

        try {

            String sourcecode = "c://programa.pas";

            PascalLexer lexer = new PascalLexer( new FileReader( sourcecode ) );

            PascalToken token;
            while ((token = lexer.yylex()) != null)  {
                System.out.println( "<" + token.nome + "," + token.valor + ">");
            }

        }
        catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```

5. Crie o arquivo **Pascal.lex**. Esse arquivo deve ter uma estrutura semelhante ao do exercício anterior. Esse arquivo Lex deve validar o arquivo fonte criado no passo 2.

```
package br.com.johnidouglas.lexicalanaliser.pascal;
import java_cup.runtime.*;

%%

%{

  private void log(String lexema, Integer linha, Integer coluna) {
         System.out.println(lexema + " Linha: " + linha + " Coluna: " + coluna );
  }

%}


%public
%class PascalLexer
%type PascalToken

%line
%column

inteiro               = 0|[1-9][0-9]*
brancos               = [\n| |\t]

%%

{inteiro} {  log(yytext(), yyline, yycolumn); return new PascalToken( "numero",
yytext()); }
{brancos} {  /**/ }

. { throw new RuntimeException("Caractere invalido \""+yytext() +
                                          "\" na linha "+yyline+", coluna
"+yycolumn); }
```

Exercícios

1. Mude o conteúdo da variável **expr** da clase **TesteAnalisadorLexico** para:

```
String expr = "if ( 2+3+a) == ( valor + 3 ) then ";
```

Adapte o arquivo **Linguagem.flex** para reconhecer os novos lexemas.

2. A linguagem de programação em questão sofre modificações e a seguinte instrução foi incluída. Mude o conteúdo da variável **expr** da clase **TesteAnalisadorLexico** para:

```
String expr = "for each( 0, 10, item, itens ) print item";
```

Adapte o arquivo **Linguagem.flex** para reconhecer os novos lexemas.

3. A linguagem de programação em questão sofre modificações e a seguinte instrução foi incluída. Mude o conteúdo da variável **expr** da clase **TesteAnalisadorLexico** para:

```
String expr = " while (parar) do { parar:=true }";
```

Adapte o arquivo **Linguagem.flex** para reconhecer os novos lexemas.

4. No arquivo **pascal.lex** implemente as regras que faltam para reconhecer todos os lexemas do arquivo fonte **programa.pas**:
   Dica: utilize as expressões regulares do analisador léxico desenvolvido no trabalho entregue.