Caneca++

Chrystian Guth
Lucas Pereira
Renan Netto

Gramática léxica

fontes/g/CanecaLexico.g

```
lexer grammar CanecaLexico;

options {
   language = Java;
}

@header {
   package br.ufsc.inf.ine5426.caneca.antlr;
}
```

Gramática léxica

```
INICIO : 'inicio' ;
FIM : 'fim' ;
SE : 'se' ;
SENAO : 'senao' ;
PARA : 'para' ;
ENQUANTO : 'enquanto' ;
TENTE : 'tente' ;
LANCE : 'lance' ;
CAPTURE : 'capture' ;
RETORNE : 'retorne' ;
```

```
CLASSE : 'classe' ;
INTERFACE : 'interface' ;
ATRIBUTO : 'atributo' ;
CONSTRUTOR : 'construtor' ;
METODO : 'metodo' ;
ESTATICO : 'estatico' ;
IMPLEMENTA : 'implementa' ;
PUBLICO : 'publico' ;
PROTEGIDO : 'protegido' ;
PRIVADO : 'privado' ;
```

Gramática léxica

```
PUBLICA : 'publica' ;
PROTEGIDA : 'protegida' ;
PRIVADA : 'privada' ;
NOVO : 'novo' ;
NOVA : 'nova' ;
ESSE : 'esse' ;
ESSA : 'essa' ;
PACOTE : 'pacote' ;
IMPORTE : 'importe' ;
```

```
CHAMADA_DE_OBJETO : '.' ;
CHAMADA_DE_CLASSE : ':' ;
PARENTESE_ESQUERDO : '(' ;
PARENTESE_DIREITO : ')' ;
COLCHETE_ESQUERDO : '[' ;
COLCHETE_DIREITO : ']' ;
ATRIBUIDOR : '=' ;
SEPARADOR : ',' ;
TERMINADOR : ';' ;
```

Gramática léxica

```
SOMA : '+' ;
SUBTRACAO : '-' ;
MULTIPLICACAO : '*' ;
DIVISAO : '/' ;
RESTO_DA_DIVISAO : '%' ;
```

```
NEGACAO : '~' ;
E : '&&' ;
OU : '||' ;
MENOR_IGUAL : '<=' ;
MAIOR_IGUAL : '>=' ;
MENOR : '<' ;
MAIOR : '>' ;
IGUAL : '==' ;
DIFERENTE : '!=' ;
```

Gramática léxica

```
VALOR_BOOLEANO : 'verdadeiro' | 'falso' ;
VALOR_NULO : 'nulo' ;
CONSTANTE_INTEIRA : ('-')? DIGITO+ ;
CONSTANTE_REAL : ('-')? DIGITO+ '.' DIGITO+ ;
LITERAL_CARACTERE : '\'' CARACTERE '\'';
LITERAL_TEXTO : '"' (CARACTERE)* '"' ;

IDENTIFICADOR : LETRA (LETRA | DIGITO | '_')* ;
IDENTIFICADOR_DE_PACOTE : '@' IDENTIFICADOR ('.'
IDENTIFICADOR)* ;

ESPACO_EM_BRANCO : (' ' | TABULACAO | QUEBRA_DE_LINHA) {skip();} ;
COMENTARIO_EM_LINHA : ('?' ~(QUEBRA_DE_LINHA)*)
{$channel=HIDDEN;} ;
COMENTARIO_EM_BLOCO : ('#' (options {greedy=false;} : .)* '#')
{$channel=HIDDEN;} ;
```

Gramática léxica

```
fragment CARACTERE : CARACTERE_DE_ESCAPE |
~(CARACTERE_NAO_IMPRIMIVEL) ;

fragment CARACTERE_NAO_IMPRIMIVEL : USADO_EM_TEXTO |
TABULACAO | QUEBRA_DE_LINHA ;

fragment CARACTERE_DE_ESCAPE : '\\' ('n' | 'r' | 't'
| 'f' | '\\' | '\'' | '"') ;

fragment QUEBRA_DE_LINHA : '\n' | '\r' ;

fragment TABULACAO : '\t' | '\f' ;

fragment USADO_EM_TEXTO : '\\' | '\'' | '"' ;

fragment LETRA : 'A'..'Z' | 'a'..'z' ;

fragment DIGITO : '0'..'9' ;
```

Analisador léxico

Através da gramática especificada o Antlr irá gerar os arquivos:

`fontes/java/.../antlr/`**`CanecaLexico.java`**
`fontes/java/.../antlr/`**`CanecaLexico.tokens`**

Compilador

Nessa primeira etapa o compilador apenas irá utilizar o analisador léxico gerado pelo Antlr para imprimir na tela os tokens reconhecidos de um dado arquivo.

`fontes/java/.../Compilador.java`

Uso do compilador:

```
java -classpath
binarios/class:bibliotecas/jar/antlr.jar
br.ufsc.inf.ine5426.caneca.Compilador.java
<arquivo.caneca>
```

Exemplos .caneca

Análise léxica dos arquivos:

`fontes/caneca/ClasseDeExemplo.caneca`
`fontes/caneca/InterfaceDeExemplo.caneca`