

INF441 - Roteiro para Trabalho Prático

Analizador Léxico usando ANTLR

Faça download do arquivo usando o link a seguir, que contém um projeto Eclipse compactado:

<https://drive.google.com/file/d/1Qmm7T5-t-1exiLGR7Y-Y8rJO-pmWhTkj/view?usp=sharing>

Em seguida, descompacte o conteúdo em uma pasta qualquer, em um computador que tenha Eclipse + ANTLR4 instalado.

Abra o IDE Eclipse e importe o projeto descompactado. Antes de iniciar este roteiro, verifique se o IDE Eclipse está devidamente configurado, seguindo as instruções em:

<https://drive.google.com/file/d/1TWBsuBXIzZmSvHniZQ80POzoCVSaPbhP/view?usp=sharing>

Em seguida, siga as instruções do roteiro para atividades relacionadas à construção de analisador léxico. Os trechos marcados em amarelo podem contar informações com mais chances de gerarem questionamentos que devem ser respondidos em arguição oral.

Diretivas Iniciais

Observe a especificação de um analisador léxico apresentado em *grammars/Lex01.g4*, escrita em formato ANTLR. A especificação inicia com

```
lexer grammar Lex01;
```

Essa declaração indica que se trata de uma especificação de um analisador léxico, cujo nome é **Lex01**. A partir dessa especificação, o ANTLR gera um analisador com o nome definido. Observe o analisador gerado como uma classe Java, na pasta *antlr-generated*. Observe também que foi gerado um arquivo adicional **Lex01.tokens**, que contém o código de cada categoria de lexema que pode ser extraído.

Regras do analisador

As regras do analisador léxico devem ter o seguinte formato:

```
identificador : definição_da_regra ;
```

Os identificadores devem sempre iniciar com letra maiúscula. Estude as regras apresentadas e procure entender o seu significado. Observe no arquivo **Lex01.tokens** os nomes dos tokens identificados na especificação, e os códigos definidos para eles.

Em seguida, abra o arquivo **Lex01.java**. Esse arquivo foi automaticamente gerado pelo ANTLR, e é gerado novamente cada vez que o arquivo **Lex01.g** é gravado. Dentro da classe Lex01, observe que um valor inteiro é associado a cada código de token:

```
public class Lex01 extends Lexer {  
    ...  
    public static final int  
        KW_CLASS=1, KW_IF=2, KW_WHILE=3, SEMICOLON=4, OPEN_PAR=5,  
        CLOSE_PAR=6, PERIOD=7, IDENT=8, INTEGER_LITERAL=9, WS=10,  
        COMMENT=11, LINE_COMMENT=12;  
    ...  
}
```

Teste do Analisador

Abra o arquivo *TesteLex.java* e observe o conteúdo:

```
import org.antlr.v4.runtime.*;

public class TesteLex {

    public static void testeLex(String fileName) throws Exception {
        ANTLRInputStream input = new ANTLRFileStream(fileName);
        Lex01 lexer = new Lex01(input);
        CommonTokenStream tokens = new CommonTokenStream(lexer);
        while (true) {
            Token t = tokens.LT(1);
            if (t.getType() == Lex01.EOF) {
                break;
            }
            System.out.println(t);
            tokens.consume();
        }

        public static void main(String args[]) throws Exception {
            testeLex("input/teste01.txt");
        }
    }
}
```

Esse arquivo contém um programa piloto que usa o analisador léxico gerado (*Lex01*) para extrair tokens de um arquivo de entrada e imprimir informações desses tokens.

Abra o arquivo *input/teste01.txt* e observe o seu conteúdo. O programa piloto irá usar esse arquivo como entrada. Execute o programa piloto *TesteLex.java* e observe a saída. Para executar, selecione o nome do programa no Package Explorer e use o menu *Run-Run*, ou então clique com o botão direito sobre o nome do programa no Package Explorer e selecione a opção *Run As - Java Application*.

Deve ser gerada uma saída como:

```
[@0,0:2='abc',<8>,1:0]
[@1,4:6='def',<8>,1:4]
[@2,9:13='while',<3>,2:0]
[@3,15:16='if',<2>,2:6]
[@4,18:22='class',<1>,2:9]
[@5,25:27='123',<9>,3:0]
[@6,64:66='456',<9>,4:0]
[@7,67:67='.',<7>,4:3]
[@8,68:69='78',<9>,4:4]
[@9,111:111='(',<5>,6:0]
[@10,112:112='a',<8>,6:1]
[@11,113:113=';',<4>,6:2]
[@12,114:114='b',<8>,6:3]
[@13,115:115=')',<6>,6:4]
```

Tente entender o que significa cada informação extraída do arquivo fonte e armazenada nos tokens construídos. Observe especialmente a definição do TIPO de cada token obtido.

Atividade 1

Na especificação *Lex01.g*, após a definição do lexeme IDENT, insira:

```
KW_PROGRAM : 'program';
```

Salve a especificação, o que irá ativar automaticamente o *builder* que executa o ANTLR, gerando um novo analisador léxico. Insira a palavra “program” em algum ponto do arquivo de entrada *teste01.txt*. Execute o programa piloto novamente e observe o código associado ao token cujo lexema é “program”.

Em seguida, mude a posição da definição KW_PROGRAM, inserindo-a antes da definição de IDENT. Salve a especificação e rode novamente o piloto. Observe que agora essa palavra deverá ser identificada com um código diferente. Procure entender esse comportamento.

Atividade 2

No arquivo de entrada *teste01.txt*, há ocorrência de uma constante de ponto flutuante (456.78) que é tratada como a sequência de 3 tokens:

```
[@6,64:66='456',<9>,4:0]  
[@7,67:67='.',<7>,4:3]  
[@8,68:69='78',<9>,4:4]
```

Insira uma regra na especificação para identificar constantes de ponto flutuante, mas de forma que o analisador continue identificando “123” como uma constante inteira. Deverão ser identificadas como constantes de ponto flutuante sequências como:

```
456.78  
.78  
456e10  
456E+5  
.78E-4
```

Teste o programa piloto com constantes de ponto flutuante no arquivo de entrada.

Instruções para entrega

Envie a especificação *Lex01.g4* alterada, usando o mecanismo de entrega de trabalhos do Google Classroom.