

Analizador Sintático

JAVA>>JAVA

Manual Do Programador

Desenvolvido Por:

Romeno João - 20181055

Gramática

Nomes e Valores

<nome> := <id> { .<id> }

<valor> := <valor_inteiro> | <valor_real> | <caracter> | <string> | null | <booleano>

Pacotes

<InicoCompilacao> := [<DeclaracaoPacote>] { <ListaImport> } { <ListaDeclaracao> }

<declaracaoPacote> := package <nome> ;

<ListaImport> := import <nome> [. *] ;

<ListaDeclaracao> := ; | <DeclaracaoClass>

Tipos

<Tipo> := <TipoPrimitivo> | <RefTipo>

<TipoPrimitivo> := byte | short | int | long | char | float | double | boolean

<RefTipo> := <VectorTipo>

<VectorTipo> := <nome> [] { [] }

Modificadores

<Modificador> := { Public | private | protected | static | abstract | final | native | synchronized | transient | volatile }

Declaração de Class

<DeclaracaoClass> := [<modificador>] class <Id> <CorpoClass>

<CorpoClass> := { [<ListaDeclaracaoCorpoClass>] }

<ListaDeclaracaoCorpoClass> := <DeclaracaoCorpoClass> { <DeclaracaoCorpoClass> }

<DeclaracaoCorpoClass> := [<modificador>] <DeclaracaoCorpoClass1>

<DeclaracaoCorpoClass1> := class <DeclaracaoClass> | <InicicalizacaoStatic> | (<tipo> | void)

<DeclaracaoMembrosClass> | <id> <DeclaracaoConstrutor>

<DeclaracaoMembrosClass> := <DeclaracaoCampo> | <DeclaracaoMetodo>

Declaração de Metodo

<DeclaracaoMetodo> := <HdrMetodo> <CorpoMetodo>

<HdrMetodo> := [<Modificador>] (<Tipo> | **void**) <DefinicaoMetodo> [<Excecao>]
<DefinicaoMetodo> := <id> ([<ListaParametrosFormais>]) { [] }
<ListaParametrosFormais> := <ParametrosFormais> { , <ParametrosFormais> }
<ParametrosFormais> := <Tipo> <IdDeclaracaoVariavel>
<Excecao> := **throws** <ListaTipoClass>
<ListaTipoClass> := <TipoClass> { , <TipoClass> }
<CorpoMetodo> := ; | <Bloco>

Declaração de variáveis e Campos

<DeclaracaoCampo> := <Tipo> <DeclaracaoCampo1>;
<DeclaracaoCampo1> := <DeclaracaoVariavel> { , <DeclaracaoVariavel> }
<DeclaracaoVariavel> := <IdDeclaracaoVariavel> [= <InicilizacaoVariavel>]
<IdDeclaracaoVariavel> := <Id> [[]]

Inicializadores

<InicilizacaoStatic> := <Bloco>
<InicializadoresVariaveis> := <InicilizacaoVariavel> [, <InicilizacaoVariavel>]
<inicilizacaoArray> := { [<InicializadoresVariaveis>] [,] }
<InicilizacaoVariavel> := <expressao> | <InicilizacaoArray>

Blocos e Declarações

<bloco> := { [<listaDeclaracoesBloco>] }
<listaDeclaracoesBloco> := <declaracoesBloco> { < declaracoesBloco > }
<declaracoesBloco> := <declaracaoVariavelLocal0> | <declaracao> | <declaracaoClass>
<declaracaoVariavelLocal0> := <declaracaoVariavelLocal> ;
<declaracaoVariavelLocal> := <tipo> <listaDeclaracaoVariavel>
<declaracao> := <declaracoes> | <declaracaoRotulo> | <declaracaoIF> | <declaracaoWhile> |
 <declaracaoFor>
<declaracoes> := <bloco> | <declaracaoVazia> | <declaracaoExpressao> | <declaracaoSwitch>
 | <declaracaoDo> | <declaracaoBreak> | <declaracaoContinue> | <DeclaracaoReturn> |

<declaracaoSynchronized> | <declaracaoThrow> | <declaracaoTry>
 <declaracaoVazia> := ;
 <declaracaoRotulo> := <id> : <declaracao>
 <declaracaoExpressao> := <Atribuicao> | <PreIncremento> | <PreDecremento> |
 <PosIncremento> | <PosDecremento> | <InvocacaoMetodo> | <InstanciarClass>
 <declaracaoIF>:= if (<expressao>) <declaracao> [else <declaracao>]
 <declaracaoSwitch> := switch (<expressao>) <BlocoSwitch>
 <BlocoSwitch> := { {<rotuloSwitch> } }
 <rotuloSwitch> := case <Expressao> : [<declaracao>] | default : [<declaracao>]
 <declaracaoWhile> := while (<expressao>) <declaracao>
 <declaracaoDo>:= do <declaracao> while (<expressao>)
 <declaracaoFor>:= for((<inicializacaoFor>; <expressao> ; <atualizacaoFor> | <tipo> <nome> :
 <nome>)) <declaracao>
 <inicializacaoFor>:= <ListaDeclaracaoExpressao> | <declaracaoVariavelLocal>
 <atualizacaoFor> := <ListaDeclaracaoExpressao>
 <ListaDeclaracaoExpressao> := <declaracaoExpressao> {<declaracaoExpressao> }
 <declaracaoBreak> := break [<id>];
 <declaracaoContinue>:= continue [<id>];
 <DeclaracaoReturn>:= return [<expressao>];

Expressões

<Expressao>:= <AtribuicaoExpressao>
 <AtribuicaoExpressao>:= <Atribuicao> | <OperacaoTernaria>
 <Atribuicao> := <LHS> <AtribuicaoOperador> <AtribuicaoExpressao>
 <LHS>:= <nome> | <FieldAccess - Duvida>| <AcederVector>
 <AtribuicaoOperador> := *= | = | /= | %= | += | -= | <=<= | >>= | >>>= | &= | ^= | |=
 <OperacaoTernaria> := <ExpressaoOU> [? <Expressao> : <OperacaoTernaria>]
 <ExpressaoOU> := <ExpressaoE> { || <ExpressaoE> }
 <ExpressaoE>:= <ExpressaoOR>{ && <ExpressaoOR> }
 <ExpressaoOR>:= <ExpressaoXOR> { | <ExpressaoXOR> }
 <ExpressaoXOR>:= <ExpressaoAND>{ ^ <ExpressaoAND> }

```

<ExpressaoAND>:= <ExpressaoIguar> { & <ExpressaoIguar> }

<ExpressaoIguar>:= <ExpressaoComparacao> { == <ExpressaoComparacao> }

<ExpressaoComparacao>:= <ExpressaoDeslocamento> { ( < > | <= | >= )
<ExpressaoDeslocamento> | instanceof <ClassInterfaceTipo> }

<ExpressaoDeslocamento>:= <ExpressaoAdicaoSubtracao> { ( << | >> | >>> )
<ExpressaoAdicaoSubtracao> }

<ExpressaoAdicaoSubtracao>:= <MultiplicacaoDivisaoMod> { ( + | - )
<MultiplicacaoDivisaoMod> }

<MultiplicacaoDivisaoMod>:= <ExpressaoUnaria> { ( * | / | % ) <ExpressaoUnaria> }

<ExpressaoUnariaSemMaisMenos> ::= <Casting> | <ExpressaoPosterior> |
~<ExpressaoUnaria> | ! <ExpressaoUnaria>

<Casting> := ( ( <Tipo> [ <ParentesesRectos> ] ) <ExpressaoUnaria> | <Expressao> )
<ExpressaoUnariaSemMaisMenos> )

<ExpressaoPosterior>:= <Principal> | <nome> | <PosIncremento> | <PosDecremento>

<PosIncremento> := <ExpressaoPosterior> ++

<PosDecremento>:= <ExpressaoPosterior> --

<ExpressaoUnaria>:= ( <PreIncremento> | <PreDecremento> |
<ExpressaoUnariaSemMaisMenos> ) { ( + | - ) ( <PreIncremento> ) | ( <PreDecremento> ) |
<ExpressaoUnariaSemMaisMenos> }

<PreIncremento> := ++ <ExpressaoUnaria>

<PreDecremento> := - - <ExpressaoUnaria>

```

Em java compilação começa a partir da declaração do pacote, logo temos o inicio de compilação a partir desse ponto do programa

Pacotes

```
<InicoCompilacao> := [<DeclaracaoPacote>] {<ListaImport>} {<ListaDeclaracao>}
```

Daí o programa inicia e vai chamando outras funções quando necessárias se o caminho específico o definir.

Tipos

<Tipo> := <TipoPrimitivo> | <VectorTipo>

<TipoPrimitivo> := **byte** | **short** | **int** | **long** | **char** | **float** | **double** | **boolean**

<VectorTipo> := <nome> **[] { [] }**

Esta parte da gramática serve para as implementações das funções que verificarão os tipos de variáveis, sejam primitivos ou vectores.

Conclusão

A tarefa da implementação do analisador sintático não foi fácil no início pois não havia percebido a parte da gramática mas posteriormente começou a revelar-se um trabalho demorado porém mais fácil.