Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
Programs		
<compilation unit> ::= <package declaration>? <import declarations>? <type declarations>?	compilation_unit : class_declaration;	O compilador considera apenas um nível de diretórios e uma classe por arquivo. Por isso tiramos o package e import e o "s" de declarations. Como os tipos resumem-se a classes mudamos o nome.
Declarations		
<package declaration> ::= package <package name> ;		Consideramos um único nível de diretórios.
<import declarations> ::= <import declaration> | <import declarations> <import declaration>		Não é possível importar módulos.
<import declaration> ::= <single type import declaration> | <type import on demand declaration>		
<single type import declaration> ::= import <type name> ;		
<type import on demand declaration> ::= import <package name> . * ;		
<type declarations> ::= <type declaration> | <type declarations> <type declaration>		No OurJava um <type declaration> resume-se a um <class declaration>, pois não existe o conceito de interface. E não é permitido mais de uma classe por arquivo.
<type declaration> ::= <class declaration> | <interface declaration> | ;		Como interface está fora do escopo do projeto, um type declaration resume-se a uma class declaration. E essa classe é obrigatória, não consideramos o vazio da gramática original.
<class declaration> ::= <class modifiers>? class <identifier> <super>? <interfaces>? <class body>	class_declaration : CLASS identifier class_body;	OurJava não tem implementa o conceito de Orientação a Objetos por isso temos uma definição mais enxuta apenas com o identificador e o corpo da classe.
<class modifiers> ::= <class modifier> | <class modifiers> <class modifier>		Visibilidade, Tipo Abstrato de Dados e Herança não estão no escopo do projeto.
<class modifier> ::= public | abstract | final		
<super> ::= extends <class type>		Herança não está no escopo do projeto.
<interfaces> ::= implements <interface type list>		Interface não está no escopo do projeto.
<interface type list> ::= <interface type> | <interface type list> , <interface type>		
	identifier : ID;	Regra inserida para realizar a comunicação com o léxico
<class body> ::= { <class body declarations>? }	<class body> ::= { <class body declaration> }	Só é permitida a declaração de uma classe por arquivo

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<class body declarations> ::= <class body declaration> | <class body declarations> <class body declaration>		
<class body declaration> ::= <class member declaration> | <static initializer> | <constructor declaration>	class_body_declaration : class_member_declaration_opt static_initializer class_member_declaration_opt;	Orientação a Objetos não está no escopo do projeto, por isso o construtor não foi especificado.
<class member declaration> ::= <field declaration> | <method declaration>	class_member_declaration_opt : class_member_declaration class_member_declaration_ | /** empty **/;	A declaração de membros da classe é opcional
	class_member_declaration_ : class_member_declaration class_member_declaration_ | /** empty **/;	
	class_member_declaration : STATIC field_or_method_declaration;	A declaração de membros da classe é opcional. Caso seja declarado deve ter static e a declaração.
	field_or_method_declaration : field_declaration | method_declaration;	
<static initializer> ::= static <block>	static_initializer : PUBLIC STATIC VOID MAIN OPEN_PAREN TYPE_STRING OPEN_COLC CLOSE_COLC ARGS CLOSE_PAREN bl	Modificamos a sintaxe para que essa regra seja usada apenas na criação do método main inicial. Dado que o mesmo requer a palavra public para compilar nos compiladores normais.
<constructor declaration> ::= <constructor modifiers>? <constructor declarator> <throws>? <constructor body>		Orientação a Objetos não está no escopo do projeto, por isso o construtor não foi especificado.
<constructor modifiers> ::= <constructor modifier> | <constructor modifiers> <constructor modifier>		
<constructor modifier> ::= public | protected | private		
<constructor declarator> ::= <simple type name> (<formal parameter list>?)		
<formal parameter list> ::= <formal parameter> | <formal parameter list> , <formal parameter>	formal_parameter_list : formal_parameter formal_parameter_list_ | formal_parameter_list_;	RDE (Recursão à Esquerda)
	formal_parameter_list_ : VIRGULA formal_parameter formal_parameter_list_ |/** empty **/;	
<formal parameter> ::= <type> <variable declarator id>	formal_parameter : primitive_type variable_declarator_id | array_type variable_declarator_id;	Em OurJava todo parametro é primitive_type ou array_type
<throws> ::= throws <class type list>		Exceção não está no escopo do projeto
<class type list> ::= <class type> | <class type list> , <class type>		Não é utilizando.
<constructor body> ::= { <explicit constructor invocation>? <block statements>? }		Orientação a Objetos não está no escopo do projeto, por isso o construtor não foi especificado.
<explicit constructor invocation>::= this (<argument list>?) | super (<argument list>?)		

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<field declaration> ::= <field modifiers>? <type> <variable declarators> ;	field_declaration :field_modifiers_ primitive_type variable_declarators PT_VIRGULA | field_modifiers_ primitive_type OPEN_COLC CLOS	Definimos desde esse nível quais os tipos de declaração de campo para evitar warnings na gramática
<field modifiers> ::= <field modifier> | <field modifiers> <field modifier>		
	field_modifiers_ : field_modifier field_modifiers_ | /** empty **/;	RDE (Recursão à Esquerda)
<field modifier> ::= public | protected | private | static | final | transient | volatile	field_modifier : FINAL | TRANSIENT | VOLATILE ;	Visibilidade não está no escopo no Projeto. Transient é uma palavra-chave para definir se um objeto é serializável e volátil se um objeto será utilizado para concorrência (ambos fora do escopo do projeto)
<variable declarators> ::= <variable declarator> | <variable declarators> , <variable declarator>	variable_declators : variable_declarator variable_declarators_;	RDE (Recursão à Esquerda)
	variable_declarators_ : VIRGULA variable_declarator variable_declarators_ | /** empty **/;	
<variable declarator> ::= <variable declarator id> | <variable declarator id> = <variable initializer>	variable_declarator : variable_declarator_id variable_declarator_ ;	RDE (Recursão à Esquerda)
	variable_declarator_ : EQUAL variable_initializer | /** empty **/;	
<variable declarator id> ::= <identifier> | <variable declarator id> [ ]	variable_declarator_id : identifier;	Uma variável só pode ser um identifier e nada mais
<variable initializer> ::= <expression> | <array initializer>	variable_initializer : assignment_expression | array_initializer | left_hand_side;	Foi acrescentado o left_hand_side para permitir várias inicializações
<method declaration> ::= <method header> <method body>	method_declaration : method_header method_body;	
<method header> ::= <method modifiers>? <result type> <method declarator> <throws>?	method_header : result_type method_declarator;	Todo método das classes deve ter um e apenas um modifier static, o qual já foi adicionado antes de chamar essa regra
<result type> ::= <type> | void	result_type : primitive_type | array_type | VOID ;	type é agora primitive_type ou array_type ou VOID
<method modifiers> ::= <method modifier> | <method modifiers> <method modifier>		Todo método das classes deve ter um e apenas um modifier static.
<method modifier> ::= public | protected | private | static | abstract | final | synchronized | native		Visibilidade está fora do escopo. Native é usado para definir um método que será definido por outra linguagem em outra compilação. As outras construções estão fora do escopo do projeto
<method declarator> ::= <identifier> (<formal parameter list>?)	method_declarator : identifier OPEN_PAREN formal_parameter_list CLOSE_PAREN;	
<method body> ::= <block> | ;	method_body : block | PT_VIRGULA | /** empty **/;	Um bloco pode não existir ou ser apenas um ponto e vírgula
<interface declaration> ::= <interface modifiers>? interface <identifier> <extends interfaces>? <interface body>		Interface está fora do escopo do projeto

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<interface modifiers> ::= <interface modifier> | <interface modifiers> <interface modifier>		
<interface modifier> ::= public | abstract		
<extends interfaces> ::= extends <interface type> | <extends interfaces> , <interface type>		
<interface body> ::= { <interface member declarations>? }		
<interface member declarations> ::= <interface member declaration> | <interface member declarations> <interface member declaration>		
<interface member declaration> ::= <constant declaration> | <abstract method declaration>		
<constant declaration> ::= <constant modifiers> <type> <variable declarator>		Foi substítuído por uma regra mais concisa.
<constant modifiers> ::= public | static | final		
<abstract method declaration>::= <abstract method modifiers>? <result type> <method declarator> <throws>? ;		Métodos abstratos estão fora do escopo do projeto
<abstract method modifiers> ::= <abstract method modifier> | <abstract method modifiers> <abstract method modifier>		
<abstract method modifier> ::= public | abstract		
<array initializer> ::= { <variable initializers>? , ? }	array_initializer : BEG variable_initializers virgula_opt END | array_creation_expression;	
	virgula_opt : VIRGULA | /** empty **/;	Regra adicionada para permitir a vírgula opcional
<variable initializers> ::= <variable initializer> | <variable initializers> , <variable initializer>	variable_initializers : variable_initializer variable_initializers_ | variable_initializers_;	RDE (Recursão à Esquerda)
	variable_initializers_ : VIRGULA variable_initializer variable_initializers_ | /** empty **/;	
<variable initializer> ::= <expression> | <array initializer>		Regra repetida na gramática original
Types		
<type> ::= <primitive type> | <reference type>		type resume-se à primitive_type
<primitive type> ::= <numeric type> | boolean	primitive_type : numeric_type | TYPE_BOOL | TYPE_STRING ;	Esses são todos os tipos permitidos

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<numeric type> ::= <integral type> | <floating-point type>	numeric_type : integral_type | floating_point_type;	
<integral type> ::= byte | short | int | long | char	integral_type : TYPE_BYTE | TYPE_SHORT | TYPE_INT | TYPE_LONG | TYPE_CHAR;	
<floating-point type> ::= float | double	floating_point_type : TYPE_FLOAT | TYPE_DOUBLE;	
<reference type> ::= <class or interface type> | <array type>	reference_type : class_type | array_type;	Interface está fora do escopo do projeto
<class or interface type> ::= <class type> | <interface type>		
<class type> ::= <type name>	class_type : identifier;	
<interface type> ::= <type name>		Interface está fora do escopo do projeto
<array type> ::= <type> [ ]	array_type : primitive_type OPEN_COLC CLOSE_COLC;	
Blocks and Commands		
<block> ::= { <block statements>? }	block : BEG block_statements END;	
<block statements> ::= <block statement> | <block statements> <block statement>	block_statements : block_statement block_statements_ | block_statements_;	RDE (Recursão à Esquerda)
	block_statements_: block_statement block_statements_ | /** empty **/;	
<block statement> ::= <local variable declaration statement> | <statement>	block_statement : local_variable_declaration_statement | statement;	
<local variable declaration statement> ::= <local variable declaration> ;	local_variable_declaration_statement : local_variable_declaration PT_VIRGULA local_variable_declaration_;	Permissão de várias declarações locais
	local_variable_declaration_ : local_variable_declaration PT_VIRGULA | /** empty **/;	
<local variable declaration> ::= <type> <variable declarators>	local_variable_declaration : primitive_type variable_declarators | primitive_type OPEN_COLC CLOSE_COLC variable_declarators;	
<statement> ::= <statement without trailing substatement> | <labeled statement> | <if then statement> | <if then else statement> | <while statement> | <for statement>	statement : statement_without_trailing_substatement | identifier TWO_POINTS statement | IF OPEN_PAREN expression CLOSE_PAREN optional_else | WHILE OPEN_PAREN expression CLOSE_PAREN statement | FOR OPEN_PAREN for_init PT_VIRGULA expression_opt PT_VIRGULA for_update_opt CLOSE_PAREN statement;	RIE (Recursão à Esquerda) com <labeled statement>, <if then statement>, <if then else statement>,<while statement> e <for statement>
<statement no short if> ::= <statement without trailing substatement> | <labeled statement no short if> | <if then else statement no short if> | <while statement no short if> | <for statement no short if>	statement_no_short_if : statement_without_trailing_substatement | identifier TWO_POINTS statement_no_short_if | IF OPEN_PAREN expression CLOSE_PAREN statement_no_short_if ELSE statement_no_short_if | WHILE OPEN_PAREN expression CLOSE_PAREN statement_no_short_if | FOR OPEN_PAREN for_init PT_VIRGULA expression_opt PT_VIRGULA for_update_opt CLOSE_PAREN statement_no_short_if;	RIE (Recursão à Esquerda) com <labeled statement no short if>, <if then else statement no short if>, <while statement no short if> e <for statement no short if>

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<statement without trailing substatement> ::= <block> | <empty statement> | <expression statement> | <switch statement> | <do statement> | <break statement> | <continue statement> | <return statement> | <synchronized statement> | <throws statements> | <try statement>	statement_without_trailing_substatement : block | empty_statement | expression_statement | switch_statement | do_statement | break_statement | continue_statement | goto_statement | return_statement;	Exceções não estão no escopo do projeto. Concorrência não está no escopo do projeto.
<empty statement> ::= ;	empty_statement : PT_VIRGULA ;	
<labeled statement> ::= <identifier> : <statement>		Referenciado em uma RIE (<statement>) - Foi retirado.
<labeled statement no short if> ::= <identifier> : <statement no short if>		Referenciado em uma RIE (<statement no short if>) - Foi retirado.
<expression statement> ::= <statement expression> ;	expression_statement : statement_expression PT_VIRGULA;	
<statement expression> ::= <assignment> | <preincrement expression> | <postincrement expression> | <predecrement expression> | <postdecrement expression> | <method invocation> | <class instance creation expression>	statement_expression : primary_no_new_array assignment_operator assignment_expression | preincrement_expression | post_incr_decrement_expression | predecrement_expression;	Não pode instanciar classes. A regra assignment foi substituida.
<if then statement>::= if (<expression>) <statement>		Referenciado em uma RIE (<statement>). Foi retirado
<if then else statement>::= if (<expression>) <statement no short if> else <statement>		Referenciado em uma RIE (<statement>). Foi retirado
<if then else statement no short if> ::= if (<expression>) <statement no short if> else <statement no short if>		Referenciado em uma RIE (<statement no short if>). Foi retirado
<switch statement> ::= switch (<expression>) <switch block>	switch_statement : SWITCH OPEN_PAREN expression CLOSE_PAREN switch_block;	
<switch block> ::= { <switch block statement groups>? <switch labels>? }	switch_block : BEG switch_block_statement_groups switch_labels END;	
<switch block statement groups> ::= <switch block statement group> | <switch block statement groups> <switch block statement group>	switch_block_statement_groups : switch_block_statement_group switch_block_statement_groups_ | switch_block_statement_groups_;	RDE (Recursão à Esquerda)
	switch_block_statement_groups_ : switch_block_statement_group switch_block_statement_groups_ | /** empty **/;	
<switch block statement group> ::= <switch labels> <block statements>	switch_block_statement_group : switch_labels block_statements;	
<switch labels> ::= <switch label> | <switch labels> <switch label>	switch_labels : switch_label switch_labels_ | switch_labels_ ;	RDE (Recursão à Esquerda)
	switch_labels_ : switch_label switch_labels_ | /** empty **/;	
<switch label> ::= case <constant expression> : | default :	switch_label : CASE expression TWO_POINTS | DEFAULT TWO_POINTS;	constant_expression é o mesmo que expression
<while statement> ::= while (<expression>) <statement>		Referenciado em uma RIE (<statement>). Foi retirado

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<while statement no short if> ::= while (<expression>) <statement no short if>		Referenciado em uma RIE (<statement no short if>). Foi retirado
<do statement> ::= do <statement> while (<expression>) ;	do_statement : DO statement WHILE OPEN_PAREN expression CLOSE_PAREN PT_VIRGULA;	
<for statement> ::= for (<for init>? ; <expression>? ; <for update>?) <statement>		Referenciado em uma RIE (<statement>). Foi retirado
<for statement no short if> ::= for (<for init>? ; <expression>? ; <for update>?) <statement no short if>		Referenciado em uma RIE (<statement no short if>). Foi retirado
<for init> ::= <statement expression list> | <local variable declaration>	for_init : statement_expression_list | local_variable_declaration | /** empty **/;	Pode não existir, ser vazia
<for update> ::= <statement expression list>	for_update : statement_expression_list;	
<statement expression list> ::= <statement expression> | <statement expression list> , <statement expression>	statement_expression_list : statement_expression statement_expression_list_ ;	RDE (Recursão à Esquerda)
	statement_expression_list_ : VIRGULA statement_expression statement_expression_list_ | /** empty **/;	
<break statement> ::= break <identifier>? ;	break_statement : BREAK identifier_opt PT_VIRGULA;	
	identifier_opt: identifier | /** empty **/;	Regra criada como forma de simular o identifier? da gramática original
<continue statement> ::= continue <identifier>? ;	continue_statement : CONTINUE identifier_opt PT_VIRGULA;	
	goto_statement : GOTO identifier_opt PT_VIRGULA;	Regra de produção adicionada para a keyword GOTO.
<return statement> ::= return <expression>? ;	return_statement : RETURN expression_opt PT_VIRGULA;	
<throws statement> ::= throw <expression> ;		Exceções estão fora do escopo do projeto.
<synchronized statement> ::= synchronized (<expression>) <block>		Concorrência está fora do escopo do projeto.
<try statement> ::= try <block> <catches> | try <block> <catches>? <finally>		Exceções estão fora do escopo do projeto.
<catches> ::= <catch clause> | <catches> <catch clause>		
<catch clause> ::= catch (<formal parameter>) <block>		
<finally > ::= finally <block>		

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
Expressions		
<constant expression> ::= <expression>		constant_expression é o mesmo que expression
<expression> ::= <assignment expression>	expression : assignment_expression;	
	expression_opt : expression | /** empty **/;	Regra de expressão opcional
<assignment expression> ::= <conditional expression> | <assignment>	assignment_expression : conditional_expression;	
<assignment> ::= <left hand side> <assignment operator> <assignment expression>		
<left hand side> ::= <expression name> | <field access> | <array access>	left_hand_side : field_access | array_access;	
<assignment operator> ::= = | *= | /= | %= | += | -= | <<= | >>= | >>>= | &= | ^= | |=	assignment_operator : EQUAL | ARITH_ASSIGN | SHIFT_ASSIGN | LOGIC_ASSIGN;	
<conditional expression> ::= <conditional or expression> | <conditional or expression> ? <expression> : <conditional expression>	conditional_expression : conditional_or_expression | OPEN_PAREN conditional_or_expression CLOSE_PAREN conditional_opt | conditional_or_expression QUESTION_MARK conditional_expression TWO_POINTS conditional_expression;	
	conditional_opt: conditional_or_expression_ | conditional_and_expression_ |inclusive_or_expression_ | exclusive_or_expression_ | and_expression_ | equality_expression_| relational_expression_ | shift_expression_ | additive_expression_ | multiplicative_expression_ | /* empty */;	
<conditional or expression> ::= <conditional and expression> | <conditional or expression> || <conditional and expression>	conditional_or_expression : conditional_and_expression conditional_or_expression_;	RDE (Recursão à Esquerda)
	conditional_or_expression_ : OR_LOGIC conditional_and_expression conditional_or_expression_ | /** empty **/;	
<conditional and expression> ::= <inclusive or expression> | <conditional and expression> && <inclusive or expression>	conditional_and_expression : inclusive_or_expression conditional_and_expression_;	RDE (Recursão à Esquerda)
	conditional_and_expression_ : AND_LOGIC inclusive_or_expression conditional_and_expression_ | /** empty **/;	
<inclusive or expression> ::= <exclusive or expression> | <inclusive or expression> | <exclusive or expression>	inclusive_or_expression : exclusive_or_expression inclusive_or_expression_ ;	RDE (Recursão à Esquerda)
	inclusive_or_expression_ : OR exclusive_or_expression inclusive_or_expression_ | /** empty **/;	
<exclusive or expression> ::= <and expression> | <exclusive or expression> ^ <and expression>	exclusive_or_expression : and_expression exclusive_or_expression_ ;	RDE (Recursão à Esquerda)
	exclusive_or_expression_ : OR_EXC and_expression exclusive_or_expression_ | /** empty **/;	
<and expression> ::= <equality expression> | <and expression> & <equality expression>	and_expression : equality_expression and_expression_ ;	RDE (Recursão à Esquerda)

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
	and_expression_ : AND equality_expression and_expression_ | /** empty **/;	
<equality expression> ::= <relational expression> | <equality expression> == <relational expression> | <equality expression> != <relational expression>	equality_expression : relational_expression equality_expression_;	RDE (Recursão à Esquerda)
	equality_expression_ : EQUALOP relational_expression equality_expression_ | /** empty **/;	
<relational expression> ::= <shift expression> | <relational expression> < <shift expression> | <relational expression> > <shift expression> | <relational expression> <= <shift expression> | <relational expression> >= <shift expression> | <relational expression> instanceof <reference type>	relational_expression : shift_expression relational_expression_;	RDE (Recursão à Esquerda)
	relational_expression_ : RELOP shift_expression relational_expression_ | /** empty **/;	
<shift expression> ::= <additive expression> | <shift expression> << <additive expression> | <shift expression> >> <additive expression> | <shift expression> >>> <additive expression>	shift_expression : additive_expression shift_expression_;	RDE (Recursão à Esquerda)
	shift_expression_ : SHIFTS additive_expression shift_expression_ | /** empty **/;	
<additive expression> ::= <multiplicative expression> | <additive expression> + <multiplicative expression> | <additive expression> - <multiplicative expression>	additive_expression : multiplicative_expression additive_expression_;	RDE (Recursão à Esquerda)
	additive_expression_ : PLUS multiplicative_expression additive_expression_ | MINUS multiplicative_expression additive_expression_ | /** empty **/;	
<multiplicative expression> ::= <unary expression> | <multiplicative expression> * <unary expression> | <multiplicative expression> / <unary expression> | <multiplicative expression> % <unary expression>	multiplicative_expression : unary_expression multiplicative_expression_ ;	RDE (Recursão à Esquerda)
	multiplicative_expression_ : MULT unary_expression multiplicative_expression_ | DIV unary_expression multiplicative_expression_ | MOD unary_expression multiplicative_expression_ | /** empty **/;	
<cast expression> ::= (<primitive type>) <unary expression> | (<reference type>) <unary expression not plus minus>	cast_expression : OPEN_PAREN primitive_type CLOSE_PAREN unary_expression | OPEN_PAREN reference_type CLOSE_PAREN unary_expression_not_plus_minus;	
<unary expression> ::= <preincrement expression> | <predecrement expression> | + <unary expression> | - <unary expression> | <unary expression not plus minus>	unary_expression : INCREMENT unary_expression | DECREMENT unary_expression | PLUS unary_expression | MINUS unary_expression | postfix_expression | NOT unary_expression | NOT_BIT unary_expression | cast_expression | OPEN_PAREN conditional_expression CLOSE_PAREN;	RIE (Recursão à Esquerda) com <predecrement expression> e <preincrement expression>
<predecrement expression> ::= -- <unary expression>	preincrement_expression : INCREMENT unary_expression;	Referenciado em uma RIE (<unary expression>). Não foi retirado pois está sendo usado em statement_expression
<preincrement expression> ::= ++ <unary expression>	predecrement_expression : DECREMENT unary_expression;	
<unary expression not plus minus> ::= <postfix expression> | ~ <unary expression> | ! <unary expression> | <cast expression>	unary_expression_not_plus_minus : postfix_expression | NOT_BIT unary_expression | NOT unary_expression | cast_expression;	Referenciado em uma RIE (<unary expression>). Não foi retirado pois está sendo usado em cast_expression
<postdecrement expression> ::= <postfix expression> --	post_incr_decrement_expression : postfix_expression;	Referenciado em uma RIE (<postfix expression>). Foi retirado.
<postincrement expression> ::= <postfix expression> ++		

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<postfix expression> ::= <primary> | <expression name> | <postincrement expression> | <postdecrement expression>	<postfix expression> ::= <primary> <postfix expression@>	RIE (Recursão à Esquerda) com <postincrement expression> e <postdecrement expression>. RDE (Recursão à Esquerda). Apagou o Expression name, pois é igual ao primary na nossa gramática
	<postfix expression@> ::= ++ <postfix expression@> | -- <postfix expression@> | E	
<method invocation> ::= <method name> (<argument list>?) | <primary> . <identifier> (<argument list>?) | super . <identifier> (<argument list>?)	method_invocation : identifier OPEN_PAREN argument_list CLOSE_PAREN | field_access OPEN_PAREN argument_list CLOSE_PAREN;	<method name> agora é identifier. Herança está fora do escopo do projeto
<field access> ::= <primary> . <identifier> | super . <identifier>	field_access : identifier POINT identifier;	Herança está fora do escopo do projeto
<primary> ::= <primary no new array> | <array creation expression>	array_access : primary_no_new_array OPEN_COLC expression CLOSE_COLC;	O array tem sua criação em separado.
<primary no new array> ::= <literal> | this | (<expression>) | <class instance creation expression> | <field access> | <method invocation> | <array access>	primary_no_new_array : LITERAL | field_access | method_invocation | array_access | identifier;	OO não é contemplado nesse projeto. Expressoes tambem retornam objetos.
<class instance creation expression> ::= new <class type> (<argument list>?)		OO não é contemplado nesse projeto.
<argument list> ::= <expression> | <argument list> , <expression>	argument_list : expression argument_list_ | argument_list_;	RDE (Recursão à Esquerda)
	argument_list_ : VIRGULA expression argument_list_ | /** empty **/;	
<array creation expression> ::= new <primitive type> <dim exprs> <dims>? | new <class or interface type> <dim exprs> <dims>?	array_creation_expression : NEW primitive_type dim_exprs dims;	OO não é contemplado
<dim exprs> ::= <dim expr> | <dim exprs> <dim expr>	dim_exprs : OPEN_COLC dim_expr_or_empty CLOSE_COLC;	Modificamos a regra e retiramos a recursão. Realizamos essa mudança para adaptar a regra ao contexto de OurJava que é mais simples.
<dim expr> ::= [<expression>]	dim_expr_or_empty : expression | /** empty **/;	
<dims> ::= [ ] | <dims> [ ]	dims: dim_exprs | /**empty**/;	
<array access> ::= <expression name> [<expression>] | <primary no new array> [<expression>]	array_access : primary_no_new_array OPEN_COLC expression CLOSE_COLC;	Simplificamos aqui também.
Tokens		
<package name> ::= <identifier> | <package name> . <identifier>		Pacotes não estão no escopo do projeto.
<type name> ::= <identifier> | <package name> . <identifier>		Resume-se a identifier que é o mesmo que <simple type name>
<simple type name> ::= <identifier>	<simple type name> ::= <identifier>	Foi para o léxico

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<expression name> ::= <identifier> | <ambiguous name> . <identifier>		<ambiguous name> foi modificado. Eh o mesmo que field access
<method name> ::= <identifier> | <ambiguous name>. <identifier>	<method name> ::= <identifier> <method name@>	Foi para o léxico
	<method name@> ::= . <identifier> | E	Foi para o léxico
<ambiguous name>::= <identifier> | <ambiguous name>. <identifier>		Consideramos que os únicos tipos existentes são os primitivos da linguagem, então não existirá recursão com o <ambiguous name>
<literal> ::= <integer literal> | <floating-point literal> | <boolean literal> | <character literal> | <string literal> | <null literal>	<literal> ::= <integer literal> | <floating-point literal> | <boolean literal> | <character literal> | <string literal> | <null literal>	Foi para o léxico
<integer literal> ::= <decimal integer literal> | <hex integer literal> | <octal integer literal>	<integer literal> ::= <decimal integer literal> | <hex integer literal> | <octal integer literal>	Foi para o léxico
<decimal integer literal> ::= <decimal numeral> <integer type suffix>?	<decimal integer literal> ::= <decimal numeral> <integer type suffix>?	Foi para o léxico
<hex integer literal> ::= <hex numeral> <integer type suffix>?	<hex integer literal> ::= <hex numeral> <integer type suffix>?	Foi para o léxico
<octal integer literal> ::= <octal numeral> <integer type suffix>?	<octal integer literal> ::= <octal numeral> <integer type suffix>?	Foi para o léxico
<integer type suffix> ::= l | L	<integer type suffix> ::= l | L	Foi para o léxico
<decimal numeral> ::= 0 | <non zero digit> <digits>?	<decimal numeral> ::= 0 | <non zero digit> <digits>?	Foi para o léxico
<digits> ::= <digit> | <digits> <digit>	<digits> ::= <digit> | <digits> <digit>	Foi para o léxico
<digit> ::= 0 | <non zero digit>	<digit> ::= 0 | <non zero digit>	Foi para o léxico
<non zero digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	<non zero digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	Foi para o léxico
<hex numeral> ::= 0 x <hex digit> | 0 X <hex digit> | <hex numeral> <hex digit>	<hex numeral> ::= 0 x <hex digit> | 0 X <hex digit> | <hex numeral> <hex digit>	Foi para o léxico
<hex digit> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | A | B | C | D | E | F	<hex digit> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | A | B | C | D | E | F	Foi para o léxico
<octal numeral> ::= 0 <octal digit> | <octal numeral> <octal digit>	<octal numeral> ::= 0 <octal digit> | <octal numeral> <octal digit>	Foi para o léxico
<octal digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	<octal digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	Foi para o léxico
<floating-point literal> ::= <digits> . <digits>? <exponent part>? <float type suffix>?	<floating-point literal> ::= <digits> . <digits>? <exponent part>? <float type suffix>?	Foi para o léxico

Original Java Syntax Specification	OurJava Syntax Specification[1]	Explicação das Mudanças Realizadas na Gramática Original de Java[2]
<digits> <exponent part>? <float type suffix>?	<digits> <exponent part>? <float type suffix>?	Foi para o léxico
<exponent part> ::= <exponent indicator> <signed integer>	<exponent part> ::= <exponent indicator> <signed integer>	Foi para o léxico
<exponent indicator> ::= e | E	<exponent indicator> ::= e | E	Foi para o léxico
<signed integer> ::= <sign>? <digits>	<signed integer> ::= <sign>? <digits>	Foi para o léxico
<sign> ::= + | -	<sign> ::= + | -	Foi para o léxico
<float type suffix> ::= f | F | d | D	<float type suffix> ::= f | F | d | D	Foi para o léxico
<boolean literal> ::= true | false	<boolean literal> ::= true | false	Foi para o léxico
<character literal> ::= ' <single character> ' | ' <escape sequence> '	<character literal> ::= ' <single character> ' | ' <escape sequence> '	Foi para o léxico
<single character> ::= <input character> except ' and \	<single character> ::= <input character> except ' and \	Foi para o léxico
<string literal> ::= " <string characters>?"	<string literal> ::= " <string characters>?"	Foi para o léxico
<string characters> ::= <string character> | <string characters> <string character>	<string characters> ::= <string character> | <string characters> <string character>	Foi para o léxico
<string character> ::= <input character> except " and \ | <escape character>		Foi para o léxico
<null literal> ::= null	<null literal> ::= null	Não aceitamos null.
<keyword> ::= abstract | boolean | break | byte | case | catch | char | class | const | continue | default | do | double | else | extends | final | finally | float | for | goto | if | implements | import | instanceof | int | interface | long | native | new | package | private | protected | public | return | short | static | super | switch | synchronized | this | throw | throws | transient | try | void | volatile | while	<keyword> ::= boolean | break | byte | case | char | class | const | continue | default | do | double | else | final | float | for | goto | if | int | long | new | return | short | static | switch | transient | void | volatile | while	Foi para o léxico

1. As regras de produção estão no formato reconhecido pelo BISON
2. Essas explicações ajudam no entendimento do escopo da linguagem OurJava