

# The Modified Grammar of the Java™ Programming Language

## 1 INTRODUCTION

This is based on chapter 18 of the book **The Java™ Language Specification, second edition** (JLS) by James Gosling, Bill Joy, Guy Steele, Gilad Bracha.

The starting goal symbol is *CompilationUnit*.

The grammar below uses the following BNF-style conventions:

\*  $[x]$  denotes zero or one occurrences of  $x$ .

\*  $\{x\}$  denotes zero or more occurrences of  $x$ .

\*  $(x \mid y)$  means one of either  $x$  or  $y$ .

\* **keyword** is an actual keyword used in the Java language.

\* Changes are recorded as additions, ~~removals~~ or

added new non-terminal:
-------------------------

rules for the added non-terminal
----------------------------------

## 2 CHANGES

The following major changes are made to the original grammar of the **The Java™ Programming Language**:

1. The new keyword **implementation** is added into the grammar for declaring an interface which includes concrete methods. The internal representation (*.class* file) of an **implementation** is the same as an **interface** except that there is a flag indicating that the declaration is an **implementation**. The distinction between the two is only at the source code level.
2. Empty declaration for **implementation** is allowed so that users can merge several **interface** and **implementation** together in just one empty declaration.
3. The new keyword **utilizes** is used to denote inheritance by a **class** from an **implementation** in parallel to the use of **implements** for inheriting from an **interface**.
4. As in the original grammar for **interface**, the keyword **extends** is used to denote inheritance of an **implementation** from another **implementation** and the keyword **implements** for denoting inheritance of an **implementation** from an **interface**.
5. **interface** is not allowed to inherit from an **implementation** because of the possible presence of code.
6. A new production rule is set up for the multi-super method invocation which takes the form of **super**(*implementationname*).*methodname*(*optionalarguments*).

### 3 THE GRAMMAR

*Identifier:*

*IDENTIFIER*

*QualifiedIdentifier:*

*Identifier* { *. Identifier* }

*Literal:*

*IntegerLiteral*

*FloatingPointLiteral*

*CharacterLiteral*

*StringLiteral*

*BooleanLiteral*

*NullLiteral*

*Expression:*

*Expression1* [*AssignmentOperator Expression1*]

*AssignmentOperator:*

=

+ =

- =

\* =

/ =

& =

| =

^ =

% =

<< =

>> =

>>> =

*Type:*

*Identifier* { *. Identifier* } *BracketsOpt*

*BasicType*

*StatementExpression:*

*Expression*

*ConstantExpression:*

*Expression*

*Expression1:*

*Expression2* [*Expression1Rest*]

*Expression1Rest:*

[ ? *Expression* : *Expression1* ]

*Expression2 :*

*Expression3* [*Expression2Rest*]

*Expression2Rest:*

$\left\{ \text{Infixop Expression3} \right\}$   
*Expression3 instanceof Type*

*Infixop:*

||  
&&  
|  
^  
&  
==  
!=  
<  
>  
<=  
>=  
<<  
>>  
>>>  
+  
-  
\*  
/  
%

*Expression3:*

*PrefixOp Expression3*  
 $\left( \left( \text{Expr} \mid \text{Type} \right) \right) \text{Expression3}$   
*Primary*  $\left\{ \text{Selector} \right\} \left\{ \text{PostfixOp} \right\}$

*Primary:*

$\left( \text{Expression} \right)$   
**this** [*Arguments*]  
**super** *SuperSuffix*  
*Literal*  
**new** *Creator*  
*Identifier*  $\left\{ . \text{Identifier} \right\} [ \text{IdentifierSuffix} ]$   
*BasicType BracketsOpt.class*  
**void.class**

*IdentifierSuffix:*

$[ \left( [ \text{BracketsOpt.class} \mid \text{Expression 1} \right) ]$   
*Arguments*  
 $\left( \text{class} \mid \text{this} \mid \text{super Arguments} \mid \text{new InnerCreator} \right)$

*PrefixOp:*

++

--  
!  
~  
+  
-

*PostfixOp:*

++  
--

*Selector:*

. *Identifier* [*Arguments*]  
**.this**  
**.super** *SuperSuffix*  
**.new** *InnerCreator*  
[ *Expression* ]

*SuperSuffix:*

*Arguments*  
[ ( *Identifier* ) ] . *Identifier* [*Arguments*]

*BasicType:*

**byte**  
**short**  
**char**  
**int**  
**long**  
**float**  
**double**  
**boolean**

*ArgumentsOpt:*

[ *Arguments* ]

*Arguments:*

( [ *Expression* { , *Expression* } ] )

*BracketsOpt:*

{ [ ] }

*Creator:*

*QualifiedIdentifier* ( *ArrayCreatorRest* | *ClassCreatorRest* )

*InnerCreator:*

*Identifier* *ClassCreatorRest*

*ArrayCreatorRest:*

[ ( [ *BracketsOpt* *ArrayInitializer* | *Expression* ] { [ *Expression* ] } *BracketsOpt* )

*ClassCreatorRest:*

*Arguments* [*ClassBody*]

*ArrayInitializer:*

{ [*VariableInitializer* { , *VariableInitializer* } [,]] }

*VariableInitializer:*

*ArrayInitializer*

*Expression*

*ParExpression:*

( *Expression* )

*Block:*

{ *BlockStatements* }

*BlockStatements:*

{ *BlockStatement* }

*BlockStatement :*

*LocalVariableDeclarationStatement*

*ClassOrInterfaceDeclaration*

[*Identifier* :] *Statement*

*LocalVariableDeclarationStatement:*

[**final**] *Type VariableDeclarators* ;

*Statement:*

*Block*

**if** *ParExpression Statement* [**else** *Statement*]

**for** ( *ForInitOpt* ; [*Expression*] ; *ForUpdateOpt* ) *Statement*

**while** *ParExpression Statement*

**do** *Statement* **while** *ParExpression* ;

**try** *Block* ( *Catches* | [*Catches*] **finally** *Block* )

**switch** *ParExpression* { *SwitchBlockStatementGroups* }

**synchronized** *ParExpression Block*

**return** [*Expression*] ;

**throw** *Expression* ;

**break** [*Identifier*]

**continue** [*Identifier*]

;

*ExpressionStatement*

*Identifier* : *Statement*

Originally missing in chapter 18, extracted from section 14.8 of JLS

*ExpressionStatement:*

*StatementExpression* ;

*Catches:*

*CatchClause* { *CatchClause* }

*CatchClause:*

**catch** ( *FormalParameter* ) *Block*

*SwitchBlockStatementGroups*:

{ *SwitchBlockStatementGroup* }

*SwitchBlockStatementGroup*:

*SwitchLabel* *BlockStatements*

*SwitchLabel*:

**case** *ConstantExpression* :

**default** :

*MoreStatementExpressions*:

{ , *StatementExpression* }

*ForInit*:

*StatementExpression* *MoreStatementExpressions*

[**final**] *Type* *VariableDeclarators*

*ForUpdate*:

*StatementExpression* *MoreStatementExpressions*

*ModifiersOpt*:

{ *Modifier* }

*Modifier*:

**public**

**protected**

**private**

**static**

**abstract**

**final**

**native**

**synchronized**

**transient**

**volatile**

**strictfp**

*VariableDeclarators*:

*VariableDeclarator* { , *VariableDeclarator* }

*VariableDeclaratorsRest*:

*VariableDeclaratorRest* { , *VariableDeclarator* }

*ConstantDeclaratorsRest*:

*ConstantDeclaratorRest* { , *ConstantDeclarator* }

*VariableDeclarator*:

*Identifier VariableDeclaratorRest*

*ConstantDeclarator:*

*Identifier ConstantDeclaratorRest*

*VariableDeclaratorRest:*

*BracketsOpt [ = VariableInitializer]*

*ConstantDeclaratorRest:*

*BracketsOpt = VariableInitializer*

*VariableDeclaratorId:*

*Identifier BracketsOpt*

**>>** *CompilationUnit:*

**[package** *QualifiedIdentifier* ; ] { *ImportDeclaration* } { *TypeDeclaration* }

*ImportDeclaration:*

**import** *Identifier* { . *Identifier* } [ . \* ] ;

*TypeDeclaration:*

*ClassOrInterfaceDeclaration*

;

*ClassOrInterfaceDeclaration:*

*ModifiersOpt* ( *ClassDeclaration* | *ImplementationDeclaration* | *InterfaceDeclaration* )

*ClassDeclaration:*

**class** *Identifier* [**extends** *Type*] [**implements** *TypeList*] [**utilizes** *TypeList*]  
*ClassBody*

*ImplementationDeclaration:*

**implementation** *Identifier* [**extends** *TypeList*] [**implements** *TypeList*]  
*ImplementationBody*

*InterfaceDeclaration:*

**interface** *Identifier* [**extends** *TypeList*] *InterfaceBody*

*TypeList:*

*Type* { , *Type* }

*ClassBody:*

{ { *ClassBodyDeclaration* } }

*ImplementationBody:*

{ { *ImplementationBodyDeclaration* } }

*InterfaceBody:*

{ { *InterfaceBodyDeclaration* } }

*ClassBodyDeclaration:*

;  
[**static**] *Block*  
*ModifiersOpt MemberDecl*

*MemberDecl:*

*MethodOrFieldDecl*  
**void** *Identifier VoidMethodDeclaratorRest*  
*Identifier ConstructorDeclaratorRest*  
*ClassOrInterfaceDeclaration*

*MethodOrFieldDecl:*

*Type Identifier MethodOrFieldRest*

*MethodOrFieldRest:*

*VariableDeclaratorRest*  
*MethodDeclaratorRest*

*InterfaceBodyDeclaration:*

;  
*ModifiersOpt InterfaceMemberDecl*

*InterfaceMemberDecl:*

*InterfaceMethodOrFieldDecl*  
~~**void** *Identifier VoidInterfaceMethodDeclaratorRest*~~  
~~**void** *Identifier VoidAbstractMethodDeclaratorRest*~~  
*ClassOrInterfaceDeclaration*

*InterfaceMethodOrFieldDecl:*

*Type Identifier InterfaceMethodOrFieldRest*

*InterfaceMethodOrFieldRest:*

*ConstantDeclaratorsRest* ;  
~~*AbstractMethodDeclaratorRest*~~  
~~*InterfaceMethodDeclaratorRest*~~

*ImplementationBodyDeclaration:*

;  
*ModifiersOpt ImplementationMemberDecl*

*ImplementationMemberDecl:*

*Type Identifier ConstantDeclaratorsRest* ;  
*Type Identifier MethodDeclaratorRest*  
**void** *Identifier VoidMethodDeclaratorRest*  
*ClassOrInterfaceDeclaration*

*MethodDeclaratorRest:*

*FormalParameters BracketsOpt* [**throws** *QualifiedIdentifierList*]  $\left( \text{MethodBody } \vdots \right)$

*AbstractMethodDeclaratorRest*

*VoidMethodDeclaratorRest:*

*FormalParameters* [**throws** *QualifiedIdentifierList*]  $\left( \text{MethodBody } \vdots \right)$

*VoidAbstractMethodDeclaratorRest*

*AbstractMethodDeclaratorRest:*

~~*InterfaceMethodDeclaratorRest:*~~

*FormalParameters BracketsOpt* [**throws** *QualifiedIdentifierList*] ;

*VoidAbstractMethodDeclaratorRest:*

~~*VoidInterfaceMethodDeclaratorRest:*~~

*FormalParameters* [**throws** *QualifiedIdentifierList*] ;

*ConstructorDeclaratorRest:*

*FormalParameters* [**throws** *QualifiedIdentifierList*] *MethodBody*

*QualifiedIdentifierList:*

*QualifiedIdentifier*  $\left\{ , \text{QualifiedIdentifier} \right\}$

*FormalParameters:*

$\left( \left[ \text{FormalParameter} \left\{ , \text{FormalParameter} \right\} \right] \right)$

*FormalParameter:*

[**final**] *Type VariableDeclaratorId*

*MethodBody:*

*Block*