Die Syntax von Java Level 1

Diese Grammatik verwendet die folgenden EBNF-Konventionen:

[x] - *x* kann einmalig oder gar nicht auftreten.

{x} - *x* kann beliebig oft (auch gar nicht) auftreten.

Alternativen stehen jeweils auf eigenen Zeilen.

Terminale sind (**blau**) in `Courier` gesetzt.

CompilationUnit:
NormalClassDeclaration

NormalClassDeclaration:
`class` *Identifier ClassBody*

ClassBody:
`{` *{ ClassBodyDeclaration }* `}`

ClassBodyDeclaration:
[Modifier] ConstructorDeclaration
[Modifier] MemberDeclaration

Modifier:
`public`
`private`
`final`

ConstructorDeclaration:
ConstructorDeclarator ConstructorBody

ConstructorDeclarator:
Identifier `(` *[FormalParameters]* `)`

ConstructorBody:
Block

MemberDeclaration:
FieldDeclaration
MethodDeclaration

FieldDeclaration:
Type VariableDeclarators `;`

VariableDeclarators:
VariableDeclarator { , VariableDeclarator }

VariableDeclarator:
Identifier [= Expression]

MethodDeclaration:
MethodHeader MethodBody

MethodHeader:
ResultType MethodDeclarator

ResultType:
Type
`void`

MethodDeclarator:
Identifier `(` *[FormalParameters]* `)`

FormalParameters:
FormalParameter { , FormalParameter }

FormalParameter:
Type Identifier

MethodBody:
Block

Block:
`{` *{ BlockStatement }* `}`

BlockStatement:
LocalVariableDeclarationStatement
Statement

LocalVariableDeclarationStatement:
Type VariableDeclarators `;`

Statement:
`if` *ParExpression Statement [* `else` *Statement]*
`return` *[Expression]* `;`
`;`
Assignment `;`
MethodInvocation `;`
Block

Assignment:
Identifier `=` *Expression*

ParExpression:
`(` *Expression* `)`

Expression:
Expression2 { InfixOperator Expression2 }

InfixOperator:
`||`
`&&`
`==`
`!=`
`<`
`>`
`<=`
`>=`
`+`
`-`
`*`
`/`
`%`

Expression2:
PrefixOperator Expression2
`(` *Type* `)` *Expression2*
Primary

PrefixOperator:
`!`
`-`

Primary:
Literal
Identifier
ClassInstanceCreationExpression
FieldAccess
MethodInvocation
ParExpression

ClassInstanceCreationExpression:
`new` *Identifier Arguments*

FieldAccess:
[Primary `.` *] Identifier*

MethodInvocation:
[Primary `.` *] Identifier Arguments*

Arguments:
`(` *[ActualParameters]* `)`

ActualParameters:
Expression { , Expression }

Type:
Identifier
`char`
`int`
`float`
`boolean`

Literal:
INTEGER_LITERAL
FLOATINGPOINT_LITERAL
BOOLEAN_LITERAL
CHARACTER_LITERAL
STRING_LITERAL
NULL_LITERAL

Identifier:
IDENTIFIER

Der syntaktische Aufbau der Literale und von *IDENTIFIER* ist bewusst nicht Teil dieser Grammatik.