Syntax

Grammar

Lexical grammar

Character classes

***LF*:** *<unicode character Line Feed U+000A>*

***CR*:**
<unicode character Carriage Return U+000D>

***WS*:**
<one of the following characters: SPACE U+0020, TAB U+0009, Form Feed U+000C>

***Underscore*:**
<unicode character Low Line U+005F>

***Letter*:**
<any unicode character from classes Ll, Lm, Lo, Lt, Lu or Nl>

***UnicodeDigit*:**
<any unicode character from class Nd>

***LineCharacter*:**
<any unicode character excluding LF and CR>

***BinaryDigit*:**
`'0'` | `'1'`

***DecimalDigit*:**
`'0'` | `'1'` | `'2'` | `'3'` | `'4'` | `'5'` | `'6'` | `'7'` | `'8'` | `'9'`

***HexDigit*:**
DecimalDigit
| `'A'` | `'B'` | `'C'` | `'D'` | `'E'` | `'F'`
| `'a'` | `'b'` | `'c'` | `'d'` | `'e'` | `'f'`

Keywords and operators

***Operator*:**
`'.'` | `','` | `'('` | `')'` | `'['` | `']'` | `'@['` | `'{'` | `'}'` | `'*'` | `'%'` | `'/'` | `'+'`
| `'-'` | `'++'` | `'--'`
| `'&&'` | `'||'` | `'!'` | `'!!'` | `':'` | `';'` | `'='` | `'+='` | `'-='` | `'*='` | `'/='` |
`'%='` | `'->'` | `'=>'`
| `'..'` | `'::'` | `'?::'` | `';;'` | `'#'` | `'@'` | `'?'` | `'?:'` | `'<'` | `'>'` | `'\m'` |
`'>='` | `'!='` | `'!=='`
| `'=='` | `'==='` | `'''` | `'"'` | `'"""'`

***SoftKeyword*:**
'public' | 'private' | 'protected' | 'internal'
| 'enum' | 'sealed' | 'annotation' | 'data' | 'inner'
| 'tailrec' | 'operator' | 'inline' | 'infix' | 'external'
| 'suspend' | 'override' | 'abstract' | 'final' | 'open'
| 'const' | 'lateinit' | 'vararg' | 'noinline' | 'crossinline'
| 'reified' | 'expect' | 'actual'

***Keyword*:**
'package' | 'import' | 'class' | 'interface'
| 'fun' | 'object' | 'val' | 'var' | 'typealias'
| 'constructor' | 'by' | 'companion' | 'init'
| 'this' | 'super' | 'typeof' | 'where'
| 'if' | 'else' | 'when' | 'try' | 'catch'
| 'finally' | 'for' | 'do' | 'while' | 'throw'
| 'return' | 'continue' | 'break' | 'as'
| 'is' | 'in' | '!is' | '!in' | 'out'
| 'get' | 'set' | 'dynamic' | '@file'
| '@field' | '@property' | '@get' | '@set'
| '@receiver' | '@param' | '@setparam' | '@delegate'

Whitespace and comments

***NL*:** *LF* | *CR* [*LF*]

***ShebangLine*:**
'#!' {*LineCharacter*}

***LineComment*:**
'//' {*LineCharacter*}

***DelimitedComment*:**
'/*' {*DelimitedComment* | <any character>} '*/'

Number literals

***RealLiteral*:**
FloatLiteral | *DoubleLiteral*

***FloatLiteral*:**
DoubleLiteral ('f' | 'F') | *DecDigits* ('f' | 'F')

***DoubleLiteral*:**
[*DecDigits*] '.' *DecDigits* [*DoubleExponent*] | *DecDigits DoubleExponent*

***LongLiteral*:**
(*IntegerLiteral* | *HexLiteral* | *BinLiteral*) 'L'

***IntegerLiteral*:**
DecDigitNoZero {*DecDigitOrSeparator*} *DecDigit* | *DecDigit*

HexLiteral:
'0' ('x'|'X') *HexDigit* {*HexDigitOrSeparator*} *HexDigit*
| '0' ('x'|'X') *HexDigit*

BinLiteral:
'0' ('b'|'B') *BinDigit* {*BinDigitOrSeparator*} *BinDigit*
| '0' ('b'|'B') *BinDigit*

DecDigitNoZero:
DecDigit - '0'

DecDigitOrSeparator:
DecDigit | *Underscore*

HexDigitOrSeparator:
HexDigit | *Underscore*

BinDigitOrSeparator:
BinDigit | *Underscore*

DecDigits:
DecDigit {*DecDigitOrSeparator*} *DecDigit* | *DecDigit*

BooleanLiteral:
'true' | 'false'

NullLiteral:
'null'

Identifiers

Identifier:
(*Letter* | *Underscore*) {*Letter* | *Underscore* | *UnicodeDigit*}
| '`' {*EscapedIdentifierCharacter*} '`'

EscapedIdentifierCharacter:
<any character except CR, LF, '`', '[', ']', '<' or '>'>

IdentifierOrSoftKey:
Identifier | *SoftKeyword*

AtIdentifier:
'@' *IdentifierOrSoftKey*

IdentifierAt:
IdentifierOrSoftKey '@'

String literals

Syntax literals are fully defined in syntax grammar due to the complex nature of string interpolation

***CharacterLiteral*:**
`'''` (*EscapeSeq* | *<any character except CR, LF, ''' and '\'>*) `'''`

***EscapeSeq*:**
UnicodeCharacterLiteral | *EscapedCharacter*

***UnicodeCharacterLiteral*:**
`'\' 'u'` *HexDigit HexDigit HexDigit HexDigit*

***EscapedCharacter*:**
`'\'` (`'t'` | `'b'` | `'r'` | `'n'` | `'` | `'"'` | `'\'` | `'$'`)

***FieldIdentifier*:**
`'$'` *IdentifierOrSoftKey*

***LineStrRef*:**
FieldIdentifier

***LineStrEscapedChar*:**
EscapedCharacter | *UnicodeCharacterLiteral*

***LineStrExprStart*:**
`'${'`

***MultiLineStringQuote*:**
`'"'` {`'"'`}

***MultiLineStrRef*:**
FieldIdentifier

***MultiLineStrText*:**
{<any character except `'"'` and `'$'`} | `'$'`

***MultiLineStrExprStart*:**
`'${'`

Misc

***EOF*:**
<end of input>

TODO: redo all the lexical grammar, right now it is a hand-written mess

Syntax grammar

***kotlinFile*:**
[*shebangLine*]
{*NL*}
{*fileAnnotation*}
packageHeader

importList
{*topLevelObject*}
EOF

***script*:**
[*shebangLine*]
{*NL*}
{*fileAnnotation*}
packageHeader
importList
{*statement semi*}
EOF

***shebangLine*:**
ShebangLine (*NL* {*NL*})

***fileAnnotation*:**
`'@file'`
{*NL*}
`':'`
{*NL*}
((`'['` (*unescapedAnnotation* {*unescapedAnnotation*}) `']'`) | *unescapedAnnotation*)
{*NL*}

***packageHeader*:**
[`'package'` *identifier* [*semi*]]

***importList*:**
{*importHeader*}

***importHeader*:**
`'import'` *identifier* [(`'.' '*'`) | *importAlias*] [*semi*]

***importAlias*:**
`'as'` *simpleIdentifier*

***topLevelObject*:**
declaration [*semis*]

***typeAlias*:**
[*modifiers*]
`'typealias'`
{*NL*}
simpleIdentifier
[{*NL*} *typeParameters*]
{*NL*}
`'='`
{*NL*}
type

***declaration*:**
classDeclaration
| *objectDeclaration*
| *functionDeclaration*
| *propertyDeclaration*
| *typeAlias*

***classDeclaration*:**
[*modifiers*]
('class' | 'interface')
{*NL*}
simpleIdentifier
[{*NL*} *typeParameters*]
[{*NL*} *primaryConstructor*]
[{*NL*} ':' {*NL*} *delegationSpecifiers*]
[{*NL*} *typeConstraints*]
[({*NL*} *classBody*) | ({*NL*} *enumClassBody*)]

***primaryConstructor*:**
[[*modifiers*] 'constructor' {*NL*}] *classParameters*

***classBody*:**
'{'
{*NL*}
classMemberDeclarations
{*NL*}
'}'

***classParameters*:**
'('
{*NL*}
[*classParameter* {{*NL*} ',' {*NL*} *classParameter*}]
{*NL*}
')'

***classParameter*:**
[*modifiers*]
['val' | 'var']
{*NL*}
simpleIdentifier
':'
{*NL*}
type
[{*NL*} '=' {*NL*} *expression*]

***delegationSpecifiers*:**
annotatedDelegationSpecifier {{*NL*} ',' {*NL*} *annotatedDelegationSpecifier*}

***delegationSpecifier*:**
constructorInvocation
| *explicitDelegation*
| *userType*
| *functionType*

***constructorInvocation*:**
userType valueArguments

***annotatedDelegationSpecifier*:**
{*annotation*} {*NL*} *delegationSpecifier*

***explicitDelegation*:**
(*userType* | *functionType*)
{*NL*}
`'by'`
{*NL*}
expression

***typeParameters*:**
`'<'`
{*NL*}
typeParameter
{{*NL*} `','` {*NL*} *typeParameter*}
{*NL*}
`'>'`

***typeParameter*:**
[*typeParameterModifiers*] {*NL*} *simpleIdentifier* [{*NL*} `':'` {*NL*} *type*]

***typeConstraints*:**
`'where'` {*NL*} *typeConstraint* {{*NL*} `','` {*NL*} *typeConstraint*}

***typeConstraint*:**
{*annotation*}
simpleIdentifier
{*NL*}
`':'`
{*NL*}
type

***classMemberDeclarations*:**
{*classMemberDeclaration* [*semis*]}

***classMemberDeclaration*:**
declaration
| *companionObject*
| *anonymousInitializer*
| *secondaryConstructor*

***anonymousInitializer*:**
'init' {*NL*} *block*

***companionObject*:**
[*modifiers*]
'companion'
{*NL*}
'object'
[{*NL*} *simpleIdentifier*]
[{*NL*} ':' {*NL*} *delegationSpecifiers*]
[{*NL*} *classBody*]

***functionValueParameters*:**
'('
{*NL*}
[*functionValueParameter* {{*NL*} ',' {*NL*} *functionValueParameter*}]
{*NL*}
')'

***functionValueParameter*:**
[*modifiers*] *parameter* [{*NL*} '=' {*NL*} *expression*]

***functionDeclaration*:**
[*modifiers*]
'fun'
[{*NL*} *typeParameters*]
[{*NL*} *receiverType* {*NL*} '.']
{*NL*}
simpleIdentifier
{*NL*}
functionValueParameters
[{*NL*} ':' {*NL*} *type*]
[{*NL*} *typeConstraints*]
[{*NL*} *functionBody*]

***functionBody*:**
block
| ('=' {*NL*} *expression*)

***variableDeclaration*:**
{*annotation*} {*NL*} *simpleIdentifier* [{*NL*} ':' {*NL*} *type*]

***multiVariableDeclaration*:**
'('
{*NL*}
variableDeclaration
{{*NL*} ',' {*NL*} *variableDeclaration*}
{*NL*}
')'

***propertyDeclaration*:**
[*modifiers*]
('val' | 'var')
[{*NL*} *typeParameters*]
[{*NL*} *receiverType* {*NL*} '.']
({*NL*} (*multiVariableDeclaration* | *variableDeclaration*))
[{*NL*} *typeConstraints*]
[{*NL*} (('=' {*NL*} *expression*) | *propertyDelegate*)]
[(*NL* {*NL*}) ';']
{*NL*}
(([*getter*] [{*NL*} [*semi*] *setter*]) | ([*setter*] [{*NL*} [*semi*] *getter*]))

***propertyDelegate*:**
'by' {*NL*} *expression*

***getter*:**
([*modifiers*] 'get')
| ([*modifiers*] 'get' {*NL*} '(' {*NL*} ')' [{*NL*} ':' {*NL*} *type*] {*NL*} *functionBody*)

***setter*:**
([*modifiers*] 'set')
| ([*modifiers*] 'set' {*NL*} '(' {*annotation* | *parameterModifier*} *setterParameter* ')' [{*NL*} ':' {*NL*} *type*] {*NL*} *functionBody*)

***setterParameter*:**
simpleIdentifier {*NL*} [':' {*NL*} *type*]

***parameter*:**
simpleIdentifier
{*NL*}
':'
{*NL*}
type

***objectDeclaration*:**
[*modifiers*]
'object'
{*NL*}
simpleIdentifier
[{*NL*} ':' {*NL*} *delegationSpecifiers*]
[{*NL*} *classBody*]

***secondaryConstructor*:**
[*modifiers*]
'constructor'
{*NL*}
functionValueParameters
[{*NL*} ':' {*NL*} *constructorDelegationCall*]

{*NL*}
[*block*]

***constructorDelegationCall*:**
('this' {*NL*} *valueArguments*)
| ('super' {*NL*} *valueArguments*)

***enumClassBody*:**
'{'
{*NL*}
[*enumEntries*]
[{*NL*} ';' {*NL*} *classMemberDeclarations*]
{*NL*}
'}'

***enumEntries*:**
enumEntry {{*NL*} ',' {*NL*} *enumEntry*} {*NL*} [',']

***enumEntry*:**
[*modifiers* {*NL*}] *simpleIdentifier* [{*NL*} *valueArguments*] [{*NL*} *classBody*]

***type*:**
[*typeModifiers*] (*parenthesizedType* | *nullableType* | *typeReference* | *functionType*)

***typeReference*:**
userType
| 'dynamic'

***nullableType*:**
(*typeReference* | *parenthesizedType*) {*NL*} (*quest* {*quest*})

***quest*:**
QUEST_NO_WS
| *QUEST_WS*

***userType*:**
simpleUserType {{*NL*} '.' {*NL*} *simpleUserType*}

***simpleUserType*:**
simpleIdentifier [{*NL*} *typeArguments*]

***typeProjection*:**
([*typeProjectionModifiers*] *type*)
| '*'

***typeProjectionModifiers*:**
typeProjectionModifier {*typeProjectionModifier*}

***typeProjectionModifier*:**
(*varianceModifier* {*NL*})
| *annotation*

***functionType*:**
[receiverType {NL} '.' {NL}]
functionTypeParameters
{NL}
'->'
{NL}
type

***functionTypeParameters*:**
'('
{NL}
[parameter | type]
{{NL} ',' {NL} (parameter | type)}
{NL}
')'

***parenthesizedType*:**
'('
{NL}
type
{NL}
')'

***receiverType*:**
[typeModifiers] (parenthesizedType | nullableType | typeReference)

***parenthesizedUserType*:**
('(' *{NL} userType {NL}* ')')
| ('(' *{NL} parenthesizedUserType {NL}* ')')

***statements*:**
[statement {semis statement} [semis]]

***statement*:**
{label | annotation} (declaration | assignment | loopStatement | expression)

***label*:**
IdentifierAt {NL}

***controlStructureBody*:**
block
| *statement*

***block*:**
'{'
{NL}
statements
{NL}
'}'

***loopStatement*:**
forStatement
| *whileStatement*
| *doWhileStatement*

***forStatement*:**
`'for'`
{*NL*}
`'('`
{*annotation*}
(*variableDeclaration* | *multiVariableDeclaration*)
IN
expression
`')'`
{*NL*}
[*controlStructureBody*]

***whileStatement*:**
(`'while'` {*NL*} `'('` *expression* `')'` {*NL*} *controlStructureBody*)
| (`'while'` {*NL*} `'('` *expression* `')'` {*NL*} `';'`)

***doWhileStatement*:**
`'do'`
{*NL*}
[*controlStructureBody*]
{*NL*}
`'while'`
{*NL*}
`'('`
expression
`')'`

***assignment*:**
(*directlyAssignableExpression* `'='` {*NL*} *expression*)
| (*assignableExpression assignmentAndOperator* {*NL*} *expression*)

***semi*:**
((`';'` | *NL*) {*NL*})
| *EOF*

***semis*:**
(`';'` | *NL* {`';'` | *NL*})
| *EOF*

***expression*:**
disjunction

***disjunction*:**
conjunction {{*NL*} `'||'` {*NL*} *conjunction*}

***conjunction*:**
equality {{*NL*} `'&&'` {*NL*} *equality*}

***equality*:**
comparison {*equalityOperator* {*NL*} *comparison*}

***comparison*:**
infixOperation [*comparisonOperator* {*NL*} *infixOperation*]

***infixOperation*:**
elvisExpression {(*inOperator* {*NL*} *elvisExpression*) | (*isOperator* {*NL*} *type*)}

***elvisExpression*:**
infixFunctionCall {{*NL*} *elvis* {*NL*} *infixFunctionCall*}

***elvis*:**
QUEST_NO_WS `':'`

***infixFunctionCall*:**
rangeExpression {*simpleIdentifier* {*NL*} *rangeExpression*}

***rangeExpression*:**
additiveExpression {`'..'` {*NL*} *additiveExpression*}

***additiveExpression*:**
multiplicativeExpression {*additiveOperator* {*NL*} *multiplicativeExpression*}

***multiplicativeExpression*:**
asExpression {*multiplicativeOperator* {*NL*} *asExpression*}

***asExpression*:**
prefixUnaryExpression [{*NL*} *asOperator* {*NL*} *type*]

***prefixUnaryExpression*:**
{*unaryPrefix*} *postfixUnaryExpression*

***unaryPrefix*:**
annotation
| *label*
| (*prefixUnaryOperator* {*NL*})

***postfixUnaryExpression*:**
primaryExpression
| (*primaryExpression* (*postfixUnarySuffix* {*postfixUnarySuffix*}))

***postfixUnarySuffix*:**
postfixUnaryOperator
| *typeArguments*
| *callSuffix*
| *indexingSuffix*
| *navigationSuffix*

***directlyAssignableExpression*:**
(*postfixUnaryExpression assignableSuffix*)
| *simpleIdentifier*

***assignableExpression*:**
prefixUnaryExpression

***assignableSuffix*:**
typeArguments
| *indexingSuffix*
| *navigationSuffix*

***indexingSuffix*:**
`'['`
{*NL*}
expression
{{*NL*} `','` {*NL*} *expression*}
{*NL*}
`']'`

***navigationSuffix*:**
{*NL*} *memberAccessOperator* {*NL*} (*simpleIdentifier* | *parenthesizedExpression* | `'class'`)

***callSuffix*:**
([*typeArguments*] [*valueArguments*] *annotatedLambda*)
| ([*typeArguments*] *valueArguments*)

***annotatedLambda*:**
{*annotation*} [*label*] {*NL*} *lambdaLiteral*

***typeArguments*:**
`'<'`
{*NL*}
typeProjection
{{*NL*} `','` {*NL*} *typeProjection*}
{*NL*}
`'>'`

***valueArguments*:**
(`'('` {*NL*} `')'`)
| (`'('` {*NL*} *valueArgument* {{*NL*} `','` {*NL*} *valueArgument*} {*NL*} `')'`)

***valueArgument*:**
[*annotation*]
{*NL*}
[*simpleIdentifier* {*NL*} `'='` {*NL*}]
[`'*'`]
{*NL*}
expression

***primaryExpression*:**
parenthesizedExpression
| *simpleIdentifier*
| *literalConstant*
| *stringLiteral*
| *callableReference*
| *functionLiteral*
| *objectLiteral*
| *collectionLiteral*
| *thisExpression*
| *superExpression*
| *ifExpression*
| *whenExpression*
| *tryExpression*
| *jumpExpression*

***parenthesizedExpression*:**
`'('`
{*NL*}
expression
{*NL*}
`')'`

***collectionLiteral*:**
(`'['` {*NL*} *expression* {{*NL*} `','` {*NL*} *expression*} {*NL*} `']'`)
| (`'['` {*NL*} `']'`)

***literalConstant*:**
BooleanLiteral
| *IntegerLiteral*
| *HexLiteral*
| *BinLiteral*
| *CharacterLiteral*
| *RealLiteral*
| *NullLiteral*
| *LongLiteral*
| *UnsignedLiteral*

***stringLiteral*:**
lineStringLiteral
| *multiLineStringLiteral*

***lineStringLiteral*:**
QUOTE_OPEN {*lineStringContent* | *lineStringExpression*} *QUOTE_CLOSE*

***multiLineStringLiteral*:**
TRIPLE_QUOTE_OPEN {*multiLineStringContent* | *multiLineStringExpression* | *MultiLineStringQuote*} *TRIPLE_QUOTE_CLOSE*

***lineStringContent*:**
LineStrText
| *LineStrEscapedChar*
| *LineStrRef*

***lineStringExpression*:**
LineStrExprStart expression `'}'`

***multiLineStringContent*:**
MultiLineStrText
| *MultiLineStringQuote*
| *MultiLineStrRef*

***multiLineStringExpression*:**
MultiLineStrExprStart
{*NL*}
expression
{*NL*}
`'}'`

***lambdaLiteral*:**
(`'{'` {*NL*} *statements* {*NL*} `'}'`)
| (`'{'` {*NL*} [*lambdaParameters*] {*NL*} `'->'` {*NL*} *statements* {*NL*} `'}'`)

***lambdaParameters*:**
lambdaParameter {{*NL*} `','` {*NL*} *lambdaParameter*}

***lambdaParameter*:**
variableDeclaration
| (*multiVariableDeclaration* [{*NL*} `':'` {*NL*} *type*])

***anonymousFunction*:**
`'fun'`
[{*NL*} *type* {*NL*} `'.'`]
{*NL*}
functionValueParameters
[{*NL*} `':'` {*NL*} *type*]
[{*NL*} *typeConstraints*]
[{*NL*} *functionBody*]

***functionLiteral*:**
lambdaLiteral
| *anonymousFunction*

***objectLiteral*:**
(`'object'` {*NL*} `':'` {*NL*} *delegationSpecifiers* {*NL*} *classBody*)
| (`'object'` {*NL*} *classBody*)

***thisExpression*:**
`'this'`
| *THIS_AT*

***superExpression*:**
('super' ['<' {*NL*} *type* {*NL*} '>'] ['@' *simpleIdentifier*])
| *SUPER_AT*

***ifExpression*:**
('if' {*NL*} '(' {*NL*} *expression* {*NL*} ')' {*NL*} (*controlStructureBody* | ';'))
| ('if' {*NL*} '(' {*NL*} *expression* {*NL*} ')' {*NL*} [*controlStructureBody*] {*NL*} [';'] {*NL*} 'else' {*NL*} (*controlStructureBody* | ';'))

***whenExpression*:**
'when'
{*NL*}
['(' *expression* ')']
{*NL*}
'{'
{*NL*}
{*whenEntry* {*NL*}}
{*NL*}
'}'

***whenEntry*:**
(*whenCondition* {{*NL*} ',' {*NL*} *whenCondition*} {*NL*} '->' {*NL*} *controlStructureBody* [*semi*])
| ('else' {*NL*} '->' {*NL*} *controlStructureBody* [*semi*])

***whenCondition*:**
expression
| *rangeTest*
| *typeTest*

***rangeTest*:**
inOperator {*NL*} *expression*

***typeTest*:**
isOperator {*NL*} *type*

***tryExpression*:**
'try' {*NL*} *block* ((({*NL*} *catchBlock* {{*NL*} *catchBlock*}) [{*NL*} *finallyBlock*]) | ({*NL*} *finallyBlock*))

***catchBlock*:**
'catch'
{*NL*}
'('
{*annotation*}
simpleIdentifier
':'
type
')'

{*NL*}
block

***finallyBlock*:**
'finally' {*NL*} *block*

***jumpExpression*:**
('throw' {*NL*} *expression*)
| (('return' | *RETURN_AT*) [*expression*])
| 'continue'
| *CONTINUE_AT*
| 'break'
| *BREAK_AT*

***callableReference*:**
[*receiverType*]
{*NL*}
'::'
{*NL*}
(*simpleIdentifier* | 'class')

***assignmentAndOperator*:**
'+='
| '-='
| '*='
| '/='
| '%='

***equalityOperator*:**
'!='
| '!=='
| '=='
| '==='

***comparisonOperator*:**
'<'
| '>'
| '<='
| '>='

***inOperator*:**
'in'
| *NOT_IN*

***isOperator*:**
'is'
| *NOT_IS*

***additiveOperator*:**
'+'

| '-'

***multiplicativeOperator*:**
'*'
| '/'
| '%'

***asOperator*:**
'as'
| 'as?'

***prefixUnaryOperator*:**
'++'
| '--'
| '-'
| '+'
| *excl*

***postfixUnaryOperator*:**
'++'
| '--'
| (*EXCL_NO_WS excl*)

***excl*:**
EXCL_NO_WS
| *EXCL_WS*

***memberAccessOperator*:**
'.'
| *safeNav*
| '::'

***safeNav*:**
QUEST_NO_WS '.'

***modifiers*:**
annotation | *modifier* {*annotation* | *modifier*}

***modifier*:**
(*classModifier* | *memberModifier* | *visibilityModifier* | *functionModifier* | *propertyModifier* | *inheritanceModifier* | *parameterModifier* | *platformModifier*)
{*NL*}

***typeModifiers*:**
typeModifier {*typeModifier*}

***typeModifier*:**
annotation
| ('suspend' {*NL*})

***classModifier*:**
'enum'

| 'sealed'
| 'annotation'
| 'data'
| 'inner'

***memberModifier*:**
'override'
| 'lateinit'

***visibilityModifier*:**
'public'
| 'private'
| 'internal'
| 'protected'

***varianceModifier*:**
'in'
| 'out'

***typeParameterModifiers*:**
typeParameterModifier {*typeParameterModifier*}

***typeParameterModifier*:**
(*reificationModifier* {*NL*})
| (*varianceModifier* {*NL*})
| *annotation*

***functionModifier*:**
'tailrec'
| 'operator'
| 'infix'
| 'inline'
| 'external'
| 'suspend'

***propertyModifier*:**
'const'

***inheritanceModifier*:**
'abstract'
| 'final'
| 'open'

***parameterModifier*:**
'vararg'
| 'noinline'
| 'crossinline'

***reificationModifier*:**
'reified'

***platformModifier*:**
    `'expect'`
    | `'actual'`

***annotation*:**
    (*singleAnnotation* | *multiAnnotation*) {*NL*}

***singleAnnotation*:**
    (*annotationUseSiteTarget* {*NL*} *unescapedAnnotation*)
    | (`'@'` *unescapedAnnotation*)

***multiAnnotation*:**
    (*annotationUseSiteTarget* {*NL*} `'['` (*unescapedAnnotation* {*unescapedAnnotation*}) `']'`)
    | (`'@'` `'['` (*unescapedAnnotation* {*unescapedAnnotation*}) `']'`)

***annotationUseSiteTarget*:**
    `'@'` (`'field'` | `'property'` | `'get'` | `'set'` | `'receiver'` | `'param'` | `'setparam'` | `'delegate'`) {*NL*} `':'`

***unescapedAnnotation*:**
    constructorInvocation
    | *userType*

***simpleIdentifier*:**
    Identifier
    | `'abstract'`
    | `'annotation'`
    | `'by'`
    | `'catch'`
    | `'companion'`
    | `'constructor'`
    | `'crossinline'`
    | `'data'`
    | `'dynamic'`
    | `'enum'`
    | `'external'`
    | `'final'`
    | `'finally'`
    | `'get'`
    | `'import'`
    | `'infix'`
    | `'init'`
    | `'inline'`
    | `'inner'`
    | `'internal'`
    | `'lateinit'`
    | `'noinline'`
    | `'open'`

| 'operator'
| 'out'
| 'override'
| 'private'
| 'protected'
| 'public'
| 'reified'
| 'sealed'
| 'tailrec'
| 'set'
| 'vararg'
| 'where'
| 'expect'
| 'actual'
| 'const'
| 'suspend'

***identifier*:**
simpleIdentifier {{*NL*} '.' *simpleIdentifier*}