Left factoring :-

Sb → [ ] <t2> func id (<P>) <n> { <n> <MST> } <sb'>
[ ] <arr> <sb'> | func id (<P>) <n> { <n> <MST>
} <sb'> | id <init> <list> <sb'>

Sb → [ ] func id (<P>) <n> { <n> <MST> } |

[,] func id (<P>) <n> { <n> <MST> } |

[ ] id <arrI> <sb'> | [,] id <arrI> <sb'>
func id (<P>) <n> { <n> <MST> } | id <init>
<list> <sb'>

Sb → [ ] <sb1> | [,] <sb1> | func -- | id --

Sb1 → func id (<P>) <n> { <n> <MST> } | id <arrI> <sb'>

c_body' → abstract <C> | sealed <C> | void fun id (<P>) { <n> <MST> } | id <C1> | dt <C> | dt <C2> <c_body'> | dt <C3>

<c_body> → dt id <init> <list> | dt [<t2> fun id (<P>) <n> { <n> <MST> } | dt fun id (<P>) <n> { <n> <MST> } |

dt [<arr> <c_body'> |

dt [<t2> fun id (<P>) { <n> <MST> } |
dt fun id (<P>) { <n> <MST> } |
dt id <init> <list>

<c_body'> → dt [<c_body1'> | dt <c_body2'>

<c_body1'> → <t2> fun id (<P>) <n> { <n> <MST> } |
<arr> <c_body'>

<c_body2'> → id <init> <list> | fun id (<P>) <n> { <n> <MST> }

S2 → static <S3> | id <C1> | dt <C2> | <sealed> | <attrs>
abstract <C> | dt <C> | dt <C3>

S2 → dt [<attrs' | dt id <init> <list> |

dt [<t2> func id (<P>) <ns> { <ns> <MST> } | dt func id(<P>) {

| dt [<t2> func id(<P>) { MST } | dt func id(P) {

| dt id <init> <list>

S2 ⟶ dt [<S2-1> | dt <S2-2> | static ...

<S2-1> ⟶ <attrs' | <t2> func id (<P>) <ns> { <ns> <MST> } |

S2-2 ⟶ id <init> <list> | func id (<P>) { MST }

<SST> ⟶ <while> | <do_while> | <for> | <if_else> |
<return> | <break> | <continue> |
int_dec <X> | id <SST1> | dt <SST2>
| void fun id (<P>) <ns> { <ns> <MSST> }

<SST2> ⟶ id <init> <list> | [<SST1>' | <SST>

{ - - - | id <SST1> | - - - }

<SST2> ⟶ id <SST2-1> | - - -

<SST2-1> ⟶ <init> <list> | <SST1>

<oe> → <ae> <oe'>

→ <re> <ae'> <oe'>

→ <e> <re'> <ae'> <oe'>

→ <T> <e'> <re'> <ae'> <oe'>

→ <F> <T'> <e'> <re'> <ae'> <oe'>

* <oe> → <const> <T'> <e'> <re'> <ae'> <oe'> <oe'>

(<e>) <T'> <e'> <re'> <ae'> <oe'> <oe'>

| <F> <T'> <e'> <re'> <ae'> <oe'> <oe'>

inc_dec <X> <T'> <e'> <re'> <ae'> <oe'> <oe'>

id <F'> <T> <e'> <re'> <ae'> <oe'> <oe'>

* <oe''> → ε

<oe'> → OR <ae> <oe'> | ε

* oe' → OR <oe> <oe'> <oe'> | <oe''>

* <oe'' → ε

<ae> → <re> <ae'>

→ <e> <re'> <ae'>

→ <T> <e'> <re'> <ae'>

→ <F> <T'> <T> <e'> <re'> <ae'>

* ae → const <T'> <T> <e'> <re'> <ae'> <ae'>

(<e>) <T'> <T> <e'> <re'> <ae'> <ae'>

| <F> <T'> <T> <e'> <re'> <ae'> <ae'>

inc_dec <X> <T'> <T> <e'> <re'> <ae'> <ae'>

<id <F' <T' <T> <e' <re' <ae' <ae'

* ae' ⟶ ε

ae' ⟶ AND <re> <ae' | ε

* ae' ⟶ AND <re> <ae' <ae'' | <ae'

* ae'' ⟶ ε

re ⟶ <e> <re' | ε

⟶ <T> <e' <re' | ε

⟶ <F> <T' <#> <e' <re' | ε

* re ⟶ <const> <T' <e' <re' <re | (<e>) <T' <e' <re' | ! <F> <T' <e' <re' <re | inc dec <X> <T' <e' <re' <re | id <F' <T' <e' <re' <re | ε

* re' ⟶ ε

re' ⟶ ROP <e> <re' | ε

* re' ⟶ ROP <e> <re' <re' | <re''

* re'' ⟶ ε

$e \longrightarrow \underline{<T>}\ <e'>$

$\longrightarrow \underline{<F>}\ <T'>\ <e'>$

$*e \longrightarrow \underline{<const>}\ <T'>\ <e'>_{<e''>} \mid \underline{(<e>)}\ <T'>\ <e'>_{<e''>} \mid \underline{!<F>}\ <T'>\ <e'>\ <e''>$

$\mid \underline{inc\_dec\ <X>}\ <T'>\ <e'>_{<e''>} \mid \underline{id<F'>}\ <T'>\ <e'>\ <e''>$

$*e' \longrightarrow \varepsilon$

$e' \longrightarrow PM\ <T>\ <e'> \mid \varepsilon$

$*e' \longrightarrow PM\ <T>\ <e'>\ <e''> \mid <e''>$

$*e'' \longrightarrow \varepsilon$

$T \longrightarrow \underline{<F>}\ <T'>$

$*T \longrightarrow \underline{<const>}\ <T'>_{<T''>} \mid \underline{(<e>)}\ <T'>_{<T''>} \mid \underline{!<F>}\ <T'>_{<T''>} \mid$

$\underline{inc\_dec\ <X>}\ <T'>_{<T''>} \mid \underline{id\ <F'>}\ <T'>\ <T''>$

$*T'' \longrightarrow \varepsilon$

$T' \longrightarrow MDM\ <F>\ <T'> \mid \varepsilon$

$*T' \longrightarrow MDM\ <F>\ <T'>\ <T''> \mid <T''>$

$*T'' \longrightarrow \varepsilon$

<F> ⟶ <const> | (<e>) | !<F> | inc-dec <X> | id <F>

* F ⟶ <const> <F'> | ('(<e>) <F'> | !<F> <F'> |
inc-dec <X> <F'> | id <F'>

* F' ⟶ ε

F' ⟶ <Z> | ε
⟶ •id <Z2> | [<oe> <W1> <Z1> | inc-dec
(<PL>) <Z2> | ε

*F' ⟶ •id <Z2> <F''> | [<oe> <W1> <Z1> <F''> |
inc-dec <F''> | (<PL>) <Z2> <F''> | <F''>

* F'' ⟶ ε

x ⟶ x

<Z> ⟶ •id <Z2> | [<oe> <W1> <Z1> | inc-dec |
(<PL>) <Z2>

*Z ⟶ •id <Z2> <Z'> | [<oe> <W1> <Z1> <Z'> | inc-de
|(<PL>) <Z2> <Z'>

* Z' ⟶ ε

~~<Z1>~~

<Z1> ⟶ •id <Z> | inc-dec

* Z1 ⟶ •id <Z> <Z1'> | inc-dec <Z1'>

* Z1' ⟶ ε

<Z2> ⟶ •id <Z> | [<oe> <W1> <Z1> | ε

* Z2 ⟶ •id <Z> <Z2'> | [<oe> <W1> <Z1> <Z2'> | ε

* Z2' ⟶ ε

$\langle W \rangle \longrightarrow$ .id $\langle W_3 \rangle$ | [$\langle oes \rangle \langle W_1 \rangle \langle W_2 \rangle$ | inc_dec | $\langle assign\_op \rangle \langle oes \rangle$ | ($\langle PL \rangle$) $\langle W_3 \rangle$

* $W \longrightarrow$ .id $\langle W_3 \rangle \langle W' \rangle$ | [$\langle oes \rangle \langle W_1 \rangle \langle W_2 \rangle \langle W' \rangle$ | inc_dec $\langle W' \rangle$ | $\langle assign\_op \rangle \langle oes \rangle \langle W' \rangle$ | ($\langle PL \rangle$) $\langle W_3 \rangle \langle W' \rangle$

* $W' \longrightarrow \varepsilon$

$W_1 \longrightarrow$] | , $\langle oes \rangle$]

* $W_1 \longrightarrow$] $\langle W_1' \rangle$ | , $\langle oes \rangle$] $\langle W_1' \rangle$

* $W_1' \longrightarrow \varepsilon$

$W_2 \longrightarrow$ .id $\langle W_3 \rangle$ | inc_dec | $\langle Assg\_op \rangle \langle oes \rangle$

* $W_2 \longrightarrow$ .id $\langle W_3 \rangle \langle W_2' \rangle$ | inc_dec $\langle W_2' \rangle$ | $\langle Assg\_op \rangle \langle oes \rangle \langle W_2' \rangle$

* $W_2' \longrightarrow \varepsilon$

$W_3 \longrightarrow$ .id $\langle W_3 \rangle$ | [$\langle oes \rangle \langle W_1 \rangle \langle W_2 \rangle$ | ε

* $W_3 \longrightarrow$ .id $\langle W_3 \rangle \langle W_3' \rangle$ | [$\langle oes \rangle \langle W_1 \rangle \langle W_2 \rangle \langle W_3 \rangle$ | $\langle W_3' \rangle$

* $W_3' \longrightarrow \varepsilon$

x ——— x

$\langle X_3 \rangle \longrightarrow id \langle X_1 \rangle$

$\langle X_3 \rangle \longrightarrow id \langle X_1 \rangle \langle X_3' \rangle$

$X' \longrightarrow \varepsilon$

$X_2 \longrightarrow \underline{\langle X_4 \rangle} \mid \underline{\langle X_1 \rangle} \mid \varepsilon$

$\longrightarrow (\langle PL \rangle) \langle X_4' \rangle \mid$
$id \langle X_2 \mid [\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle \mid$
ε

$X_2 \longrightarrow (\langle PL \rangle) \langle X_4' \rangle \langle X_2' \rangle \mid id \langle X_2 \rangle \langle X_2' \rangle \mid [\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle$
$\langle X_2' \rangle \mid \langle X_2' \rangle$

$X_2' \longrightarrow \varepsilon$

$X_3 \longrightarrow [\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle$

$X_3 \longrightarrow [\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle \langle X_3' \rangle$

$X_3' \longrightarrow \varepsilon$

$X_4 \longrightarrow (\langle PL \rangle) \langle X_4' \rangle$

$X_4' \longrightarrow \varepsilon$

$X_4' \longrightarrow \underline{\langle X_3 \rangle} \mid \langle X_3 \rangle$

$X_4' \longrightarrow [\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle \langle X_4' \rangle \mid id \langle X_1 \rangle \langle X_4' \rangle$

$X_4'' \longrightarrow \varepsilon$

$\langle X_1 \rangle \longrightarrow id \langle X_2 \rangle \mid \underline{\langle X_3 \rangle} \mid \varepsilon$

$\longrightarrow id \langle X_2 \rangle \mid \varepsilon \mid [\langle oe \rangle$
$\langle W_1 \rangle \langle X_1 \rangle$

$\langle X_1 \rangle \longrightarrow id \langle X_2 \rangle \langle X_1' \rangle \mid \langle X_1' \rangle \mid$
$[\langle oe \rangle \langle W_1 \rangle \langle X_1 \rangle \langle X_1' \rangle$

$X_1' \longrightarrow \varepsilon$

<fun_call> ⟶ id <Y>

fun_call ⟶ id <Y> <fun_call'>

<fun_call>' ⟶ ε

Assign_st ⟶ <X> <Assign_op> <oe>

⟶ id <X1> <Assig_op> <oe> <Assign_st>

<Assign_st>' ⟶ ε

<inc_dec_st> ⟶ <X> inc_dec | inc_dec <X>

⟶ id <X1> inc_dec <inc_dec_st>' | inc_dec <X> <inc_dec>'

<inc_dec_st>' ⟶ ε

x ⟶ x

Y ⟶ .id <Y> | (<PL>) <Y1> | <a> .id <Y>

Y ⟶ .id <Y> <Y'> | (<PL>) <Y1> <Y'> | [<oe> <W1> . id <Y> <Y'>

Y' ⟶ ε

Y1 ⟶ .id <Y> | <a> .id <Y> | ε

Y1 ⟶ .id <Y> <Y1>' | [<oe> <W1> .id <Y> <Y1'> |Y1'>

Y1' ⟶ ε

<a> → [<oes> <W1> <a'>

<a> → ε

x ← x

<dec> → dt id <init> <list> | id id <init> <list>
<dec'> <dec'>

<dec'> → ε

list → , id <init> <W1> <list'> | ε list'

list' → ε

init → = <oes> | ε init'
<init'>

init' → ε

x ← x

func_def → <static> <types> func id (<P>) <n>
{ <n> <MST> }

→ static <types> func id (<P>) <n> { <n> <MST> }

~~<types>~~ <types> func id (<P>) <n> { <n> <MST> }

→ static <types> func id (<P>) <n> { <n> <MST> }
dt <type'> func id (<P>) <n> { <n> <MST> } |
dt <type'> func id (<P>) <n> { <n> <MST> }

id <type'> fun id (<P>) <n> { <n> <MST> }

fun_def ⟶ static <type> fun id (<P>) <n> {
<n> { <n> <MST> } <fun_def'> |
dt <type'> fun id (<P>) <n> { <n> <MST>
<fun_def'> | void fun id (<P>) <n> { <MST> } <fun_def'> |
id <type'> fun id (<P>) <n> { <n> <MST>
<fun_def'>

<fun_def'> ⟶ ε

<static> ⟶ static | ε
static ⟶ static <static'> | <static'>
<static'> ⟶ ε

type ⟶ dt <type'> | void | id <type'>
type ⟶ dt <type'> <type''> | void <type'> |
id <type'> <type''>
type'' ⟶ ε

type' ⟶ [<t2> | ε <type''>
type'' ⟶ ε
t2 ⟶] <t2'> | ,] <t2>

P ⟶ dt <P1> <P'> | id <P1> <P''> | ε <P'>

P'' ⟶ ε

P1 ⟶ id <P1'> <P3> <P1''>

P1'' ⟶ ε

P1' ⟶ [<P2> <P1''> | ε <P1''>

P1'' ⟶ ε

P' ⟶ id <P1> <P''> | dt <P1> <P''>

P2 ⟶] <P3> <P2'> | ,] <P3> <P2'>

<P2'> ⟶ ε

P3 ⟶ , <P'> <P3'> | ε <P3'>

X ⟷ X

<type'> ⟶ [<t2> <type''> | ε <type''>

<type''> ⟶ ε

<t2> ⟶] <t2'> | ,] <t2'>

t2' ⟶ ε

X ⟷ X

<arr_def> → id <arr> <arr_def'> | dt <arr> <arr_def'>

<arr_def'> → ε

<arr> → [<arr'> <arr'>

arr'' → ε

arr' →] id <arr1> <arr''> | ,] id <arr1> <arr''>

arr'' → ε

arr1 → = <arr1'> <arr1''> | ε <arr1''>

arr1'' → ε

arr1' → new <arr2> <arr1''> | { <values> } <arr1''>

<arr1''> → ε

arr2 → dt [<oe> <arr3> <arr2'> | id [<oe> <arr3> <arr2'>

<arr2'> → ε

arr3 →] { <values> } <arr3'> | , <oe>] { <values> } <arr3'>

<arr3'> → ε

values ⟶ <oes> <V>

⟶ <ae> <oe'> <V>

⟶ <re> <ae'> <oe'> <V>

⟶ <es> <re'> <re> <ae'> <oe'> <V>

⟶ <T> <e'> ~~<e>~~ <re'> <re> <ae'> <oe'> <V>

⟶ <F> <T'> <e'> <re'> <re> <ae'> <oe'> <V>

values ⟶ <const> <T'> <e'> <re'> <re> <ae'> <oe'> <values'> <B> |

(<es>) <T'> <e'> <re'> <re> <ae'> <oe'> <V> |

!<F> <T'> <e'> <re'> <re> <ae'> <oe'> <V> |

inc_dec <X> <T'> <e'> <re'> <re> <ae'> <oe'> <values'> <B> |

id <F'> <T'> <e'> <re'> <re> <ae'> <oe'> <V>

<values'>

<values'> ⟶ ε

*values' ⟶ { <values> } <V> <values'>

values' ⟶ ε

* V ⟶ , <oes> <V> <B> | ε <V'>

<V'> ⟶ ε

<values'> ⟶ <values'> <V> | ε <V'>

* <break_st> ⟶ break <break_st'> | ε <break_st'>

<break_st'> ⟶ ε

* <continue_st> ⟶ continue <continue_st'> | ε <continue_st'>

<continue_st'> ⟶ ε

* return_st ⟶ return <r'> β

<return_st> ⟶ return <r'> <return_st'>

<return_st'> ⟶ ε

* <r'> ⟶ <oes> < | ε

⟶ <aes> <oes'> | ε

⟶ <res> <aes'> <oes'> | ε

⟶ <e> <res'> <aes'> <oes'> | ε

⟶ <T> <e'> <res'> <aes'> <oes'> | ε

⟶ <F> <T'> <e'> <res'> <aes'> <oes'> | ε

<res'> ⟶ <const> <T'> <e'> <res'> <aes'> <oes'> | (<e>) <T'>
<e'> <res'> <aes'> <oes'> | ! <F> <T'> <e'> <res'>
<aes'> <oes'> | inc_dec <X> <T'> <e'> <res'>
<aes'> <oes'> | id <F'> <T'> <e'> <res'> <aes'> <oes'>

ε

* <n> → \n <n'> <n''>

<n''> → ε

* <n'> → <n> <n''> | ε <n'>

<n'''> → ε

* <MST> → ε <MST'> | <SST> <n> <MST> <MST'>

<MST'> → ε

* <if-else> → if <oe> : <n> <body> <op-else> <if-else'>

<if-else'> → ε

* body → ε <body'> | <SST> <n> <body'> | { <n> <MST> } <n> <body'>

<body'> → ε

* <op-else> → else : <n> <body> <op-else'> | ε <op-else'>

<op-else'> → ε

* <while_st> → while <oe> : <n> <body> <while_st'>

<while_st'> → ε

* <do-while> → do <n> { <n> <MST> } <n>
while <oe> : <n> <do-while'>

<do-while'> → ε

<for> → <e1> : <e2> : <e3> <n> <body>

→ <dec> : <e2> : <e3> <n> <body> |
<Assign_st> : <e2> : <e3> <n> <body> |
: <e2> : <e3> <n> <body>

<for> → dt id <init> <list> : <e2> : <e3> <n> <body> <for'> |
id id <init> <list> : <e2> : <e3> <n> <body> <for'> |
: <e2> : <e3> <n> <body> <for'> |

<for'> → ε | <X> <Assign_op> <oe> : <e2> : e3> <n> <body>
[id <X1>] <Assign_op> <oe> : <e2> : <e3> <n>

<e1> → <dec> | <Assign> <body>.

<for> → dt id <init> <list> : <e2> : <e3> <n>
<body> <for'> | id id <init> <list> : <e2>
: <e3> <n> <body> <for'> | : <e2> : <e3> <n>
<body> <for'> | id <X1> <Assi_op> <oe>
: <e2> : <e3> <n> <body> <for'>

<for'> → ε

*Left factor :-

for → id <for1> | dt - - | : <e2> - - -
<for1> → id <init> <list> : <e2> : <e3> <n> <body> <for'> |
<X1> <Ass_op> <oe> : <e2> : <e3> <n>
<body> <for'>

left factor

e1 → id <e11> | dt ... | <e1'>

e11 → id <init> <list> <e1'> | <X1> <Assign_op>

<e1> → <decl> | <Assign_st> | ε

→ dt id <init> <list> | id id <init> <list> | <X1> <Assign_op> <oe> { [id <X1>] dtype

e1 → dt id <init> <list> <e1'> | id id <init> <list> <e1'> | id <X1> <Assign_op> <oe> <e1'> | ε

<e1'> → ε

left fact:

e2

<e2> → <oe> | ε

→ <ae> <oe'> | ε

→ <re> <ae'> <oe'> | ε

→ <e> <re'> <ae'> <oe'> | ε

→ <T> <e'> <re'> <ae'> <oe'> | ε

→ <F> <T'> <e'> <re'> <ae'> <oe'> | ε

e2 → <consts> <T'> <e'> <re'> <ae'> <oe'> <e2'> | (<e>) <e2'> | <T'> <e'> <re'> <ae'> <oe'> <e2'> | !<F> <T'> <e'> <re'> <ae'> <oe'> <e2'> | inc_dec <X> <T'> <e'> <re'> <ae'> <oe'> <e2'> | id <F'> <T'> <e'> <re'> <ae'> <oe'> <e2'> | ε

Left factor

SST → id <SST11> | dt <SST12> | - - -

SST11 → <SST1> | id <init> <list> | <X1> <Ass_op> <oes>

SST12 → <SST2> | id <init> <list>

e3 → id <e3'> <e3''> | inc_dec <X> <e3''> | ε <e3''>

<e3''> → ε

e3' → assign_op <e3'''> | inc_dec <e3'''>

<e3'''> → ε

X → X

<SST> → <while_st> | <do_while> | <if_else> | <for> |
<return_st> | <break_st> | <continue_st> |
inc_dec <X> | id <SST1> | dt <SST2>
void fun id (<P>) <n> { <n> <MST> }

→ while <oes> : <n> <body> | do <n> { <n> <MST> }
<n> while <oes> : <n> | if <oes> : <n> <body>
<op_else> | <e1> : <e2> : <e3> <n> <body> |
return <r> | break | continue | ε |
inc_dec <X> | id <SST1> | dt <SST2> |
void fun id (<P>) <n> { <n> <MST> }

→ while <oes> : <n> <body> <SST> | do <n> { <n> <MST> }
<n> while <oes> : <n> <SST> | if <oes> : <n> <body>
<op_else> <SST> | return <res> | break | continue | ε |
inc_dec <X> | id <SST1> | dt <SST2> |
void fun id (<P>) <n> { <n> <MST> } |
dt id <init> <list> | id id <init> <list>
id <op> <oes>

Left factoring

SST1 → [<SST11> | ---

SST11 → <oes> <W1> <W2> <SST1'> | <SST1'>

<SST'> → ε

SST1 → <W> | [<SST1'> | id <SST3> | <SST5>

→ id <W> | [<oes> <W1> <W2> | inc_dec |

<assig-op> <oes> | (<P2>) <W3> | [<SST1'> |

id <SST3'> | func id (<P>) <ns> { <ns>

<MST> } <SST6'>

<SST1'> → ε

SST1' →] <SST4> <SST1'> | .] <SST4> <SST1'>

SST1'' → ε

SST2 → id <init> <list> | [<SST1'> | <SST

SST2 → id <init> <list> <SST2'> | [<SST1'> <SST2'> |

while <oes> : <ns> <body> <SST2'> | do <body>

<ns> while <oes> : <ns> <SST2'> | if <oes>

<ns> <body> <op-else> <SST2'> |

Return <res'> <SST2'> | break <SST2'> |

continue <SST2'> | ε <SST2'> |

inc_dec <X> <SST2'> | id <SST1> <SST2'> |

dt <SST2> <SST2'> | void func id (<P>

<ns> { <ns> <MST> } | dt id <init> <list>

<SST2'> | id id <init> <list> <SST2'> |

id <X> <Assops> <oes> <SST2'>

<SST2'> → ε

SSTB → = <SSTB'> <SSTB>

<SSTB'> → ε

SSTB' → new id <Y> | <oes> [<aes> <oes'>
<ne> <aes'> <oes'>
<e> <ne'> <aes'> <oes'>
<T> <e'> <ne'> <aes'> <oes'>
<F> <T'> <e'> <ne'> <aes'> <oes'>

SSTB' → new id <Y> <SSTB'> | <const> <T'> <e'> <ne'> <aes'> <oes'> | (<e>) <T'> <e'> <ne'> <aes'> <oes'> <SSTB''> | ! <F> <T'> <e'> <ne'> <aes'> <oes'> <SSTB''> | inc_dec <X> <T'> <e'> <ne'> <aes'> <oes'> <SSTB''> | id <F> <T'> <e'> <ne'> <aes'> <oes'> <SSTB''>

SSTB'' → ε

SSTA → id <arr I> | <SSTS>
→ id <arr I> <SSTA'> | fun id (<P>) <n> { <n> <MST> } <SSTA'>

<SSTA'> → ε

SSTS → fun id (<P>) <n> { <n> <MST> } <SSTS'>

SSTS' → ε

<defs> → Public <defs'> | <defs'> | ε
{ <class_def> <n> <defs> | <struct_def>
<n> <defs>

<defs> → Public <defs'> | ε <defs'> ~~<abstract>~~
abstract <static> class id <inh> <n> { <n>
<c_body> } <n> <defs> <defs'> |
sealed <static> class id <inh> <n> { <n>
<c_body> } <n> <defs> <defs'> |
static class id <inh> <n> { <n> <c_body>
<n> ~~<def>~~ <c_body> } <defs'> | class id
<inh> <n> { <n> <c_body> } <n> <defs>
<defs'> | struct id <n> { <n> <sbody>
} <n> <defs> <defs'>

defs' → ε

<defs> ⟶ <class_defs> <n> <defs> |
<struct_defs> <n> <defs>

⇒ { <ab_sealed> <static> class id <inh> <n> { <n>
<c_body> } <n> <defs> |
[abstract / sealed / static / class]

⇒ { struct id <n> { <n> <sbody> } <n> <defs>

<defs> ⟶ abstract <static> class id <inh> <n> { <n>
<c_body> } <n> <defs> <defs>" | sealed <static>
class id <inh> <n> { <n> <c_body> } <n> <defs>
<defs> | static class id <inh> <n> { <n>
<c_body> } <n> <defs> <defs>" | class id
<inh> <n> { <n> <c_body> } <n> <defs> <defs>"
| struct id <n> { <n> <sbody> } <n> <defs>
<defs>"

defs" ⟶ ε

S → Public class id <inits> { <ns> <S1> <S'> |
class id <inits> { <ns> <S1> <S'>

S' → ε

S1 → Public <S2> | Private <statics> <c_body> <ns> <c_body> |
Private Protected <statics> <c_body> <ns> <c_body>
| <statics> <c_body> <ns> <c_body>

O → Public <S2> <S1'> | Private <statics> <c_body> <ns>
<c_body> <S1'> | Protected <statics> <c_body> <ns>
<c_body> <S1'> | static <c_body> <ns> <c_body> <S1'>
| <c_body> <n> <c_body> <S1'>
{ ↓ abstract / sealed / ε / \n <ns>

S1 → Public <S2> <S1'> | Private <statics> <c_body> <ns>
<c_body> <S1'> | Protected <statics> <c_body> <ns>
<c_body> <S1'> | static <c_body> <ns> <c_body>
<S1'> | abstract <ns> <c_body> <S1'> | sealed <ns>
<c_body> <S1'> | \n <ns> <c_body> <S1'>

S1' → ε

S2 → static <S3> | <c_bodys> <n> <c_bodys>

→ static <S3> <S2'> | <ab-sealed> <C> | id <C1> <S2'>

dt <C2> <S2'>

{

abstract <C> | sealed | ε <C> }

{

dt <C'> | dt <C3> | void fun id (<B>) {<n> <MST>

S2 → static <S3> <S2'> | id <C1> <S2'> | dt <C2> <S2'> |

sealed <S2'> | abstract <C> <S2'> | dt <C'> <S2'> | dt <C3> <S2'>

| void fun id (<P>) {<n> <MST>}

<S2'> → ε

S3 → void <S4> <S5> | dt <C> <S5'> | id <C3> <S3'> |
abstract <S5> <S3'>

S3' → ε

S4 → main() { <n> <MST> } <n> <C_body> } <n>
<defs> <S4'> | func id (<P>) { <n> <MST> } <S4'>

S4' → ε

<class_defs> → <ab_sealed> <static> class id <init> <n> { <n> <c_body> }
<n> { <n> <c_body> }
abstract | sealed | ε

{ abstract <static> class id <init> <n> { <n> <c_body> }
sealed "
<static> [static | ε]

{ static class id <init> <n> { <n> <c_body>
class id <init> <n> { <n> <c_body> }

<class_def> → abstract <static> class id <init> <n>
{ <n> <c_body> } <class_defs'> | sealed <static>
class id <init> <n> { <n> <c_body> } |
static class id <init> <n> { <n> <c_body>
<class_def> | class id <init> <n> { <n>
<c_body> } <class_defs'>

<c_body> → <acc_mods> <statics> <c_body> <ns>
<c_body> | ε
Public / Protected / default / Private / ε
~~Public static~~ ε / abstract / sealed / ε /
id / dt / dt / id / void

<c_body> → Public <static> <c_body> <ns> <c_body>
Protected <static> <c_body> <ns> <c_body>
default <static> <c_body> <ns> <c_body>
Private <static> <c_body> <ns> <c_body>
[static <c_body> <ns> <c_body>
{ abstract <ns> <c_body>
sealed <ns> <c_body> |
ε <ns> <c_body>
[id <C1> <ns> <c_body>
dt <C2> <ns> <c_body>
\n <ns> <c_body>

<c_body> → Public <static> <c_body> <ns> <c_body> <c_body>
Protected <static> <c_body> <ns> <c_body> <c_body>
default <static> <c_body> <ns> <c_body> <c_body>
Private <static> <c_body> <ns> <c_body> <c_body>
static <c_body> <ns> <c_body> <c_body>

abstract <n> <c_body> <c_body'> |
sealed <n> <c_body> <c_body'> |
Vn <n> <c_body> | id <C1> <n> <c_body>
<c_body'> | dt <C2> <n> <c_body> <c_body'>

c'body' ⟶ ε

c_body' ⟶ <abstract> <C> | id <C1> | dt <C2>

{ abstract <C> | sealed <C> | ε <C>

{ dt <C> | dt <C3> | void
func id (<P>) { <n>
<MST> }

c_body' ⟶ abstract <C> | sealed <C> | dt <C> | dt <C3> |
void func id (<P>) { <n> <MST> } |
id <C1> | dt <C2> <c_body'>

c_body' ⟶ ε

C ⟶ dt <C> <C'> | id <C3> <C'> | void func id (<P>)
{ <n> <MST> } <C'>

C' ⟶ ε

C → id <init> <list>
<ns> <MST> } | <type> func id (<P>) {

[<t2> | ε

C' → id <init> <list> <C''> | [<t2> func id (<P>) {
<ns> <MST> } <C3> | func id (<P>) {<ns>
<MST> } <C''>

C'' → ε

C1 → <arr> | (<P>) {<ns> <cons_body>} |
id = new id ;

[<arr'>

C1 → [<arr'> <C1'> | (<P>) {<ns> <cons_body>} <C1'>
| id = new id ; <C1'>

C1' → ε

C2 → <arr>

→ [<arr'> <C2'>

C2' → ε

C3 ⟶ types' func id (<P>) { <ns <MST> }
[<t2> | ε
| id <inits> <lists>

C3 ⟶ [<t2> func id (<P>) { <ns> <MST> } <C3'> |
func id (<P>) { <ns> <MST> } <C3'> | id <init>
<lists> <C3'>

C3' ⟶ ε

init ⟶ : id | ε <inits'>
ε
<init'>

init' ⟶ ε
ab-sealed ⟶ abstract <ab-seal'> | sealed <ab-seal'> | ε <ab-seal'>
<ab-sealed'> ⟶ ε
static ⟶ static <statics'> | ε <statics'>
<statics'> ⟶ ε

X ⟶ X

<struct_def> ⟶ struct id <n> { <n> <sbody> }
<struct_def>

<struct_def'> ⟶ ε

<sbody> ⟶ <av> <sbody 1>
[Public / Private / ε]

<sbody 1> ⟶ { <struct_def> |
<static_sbody>
<n> <sbody> }

S-body ⟶ Public <sbody 1> <s-body'> | Private <sbody 1>
<s_body'> | <struct_def> | <static> <sbody>

{ struct id <n> { <n>
<sbody> }

{ <n> <sbody>
{ static <sbody>
<n> <sbody> }
void fun id
(<P>) <n> { |
dt <sb> | ...

S-body ⟶ Public <sbody> <s_body'> | Private <sbody>
<sbody'> | struct id <n> { <n> <sbody> } <sbody'>
| static <sbody'> <n> <sbody'> | void fun id
(<P>) <n> { <n> <ST> } <sbody'> | dt <sb>
<s_body'> | id <b1> &sbody'
ε

* s_body1 → <struct_defs> <s_bodys> | <static> <sbodys'> <ns>

→ struct id <ns> { <ns> <sbodys> } <sbody1's> |
static <sbodys'> <ns> <sbodys> <sbody1's> |
void func id (<Ps>) <ns> { <ns> <MSTs> } <ns>
<s_bodys> <sbodys'> | dt <sb> <ns> <sbodys>
<sbody1's>

sbody1' → ε

* sbody' → void func id (<Ps>) <ns> { <ns> <MSTs> } <sbodys'> |
dt <sb> <sbodys'> | id <sb1> <sbodys'>

sbody'' → ε

* sb → <types'> func id (<Ps>) <ns> { <ns> <MSTs> } |
[<t2>/ε <cons> | id <inits> <lists>

[<cons'>

→ [<t2> func id (<Ps>) <ns> { <ns> <MSTs> } <sb'> |
func id (<Ps>) <ns> { <ns> <MSTs> } <sb'> |
[<cons'> <sb'> | id <inits> <lists> <sb'>

sb' → ε

* ~~stat~~ ac → Public <ac'> | Private <ac'> | ε <ac>