

$\langle \text{break-st} \rangle \rightarrow \text{break} \mid \epsilon$

$\langle \text{continue-st} \rangle \rightarrow \text{continue} \mid \epsilon$

$\langle \text{return-st} \rangle \rightarrow \text{return} \langle x' \rangle$

$\langle x' \rangle \rightarrow \langle \text{var} \rangle \mid \epsilon$

$\langle \text{ns} \rangle \rightarrow \backslash n \langle \text{ns}' \rangle$

$\langle \text{ns}' \rangle \rightarrow \langle \text{ns} \rangle \mid \epsilon$

$\langle \text{MST} \rangle \rightarrow \epsilon \mid \langle \text{ss} \rangle \langle \text{ns} \rangle \langle \text{MST} \rangle$

$\langle \text{if\_else} \rangle : \text{if } \langle \text{oe} \rangle : \langle \text{ns} \rangle \langle \text{body} \rangle \text{. if\_else}$   
 $\langle \text{body} \rangle \rightarrow \epsilon \mid \langle \text{ss} \rangle \langle \text{ns} \rangle \mid \{ \langle \text{ns} \rangle \langle \text{MST} \rangle \} \langle \text{ns} \rangle$   
 $\langle \text{op\_else} \rangle \rightarrow \text{else } \langle \text{ns} \rangle \langle \text{body} \rangle \mid \epsilon$

$\langle \text{while\_st} \rangle \rightarrow \text{while } \langle \text{oe} \rangle : \langle \text{ns} \rangle \langle \text{body} \rangle$   
 $\langle \text{do\_while} \rangle \text{ do } \langle \text{ns} \rangle \{ \langle \text{ns} \rangle \langle \text{MST} \rangle \} \langle \text{ns} \rangle$   
 $\text{while } \langle \text{oe} \rangle : \langle \text{ns} \rangle$

for  $e_1 : e_2 : e_3 \langle \text{ns} \rangle \langle \text{body} \rangle$

$\langle e_1 \rangle \rightarrow \langle \text{dec} \rangle \mid \langle \text{Assign-st} \rangle \mid \epsilon$

$\langle e_2 \rangle \rightarrow \langle \text{oe} \rangle \mid \epsilon$

$\langle e_3 \rangle \rightarrow \text{id } \langle e_3' \rangle \mid \text{inc-dec } \langle x \rangle \mid \epsilon$

$\langle e_3' \rangle \rightarrow \langle \text{Assign-ops} \rangle \mid \text{inc-dec}$

$\langle \text{arr-def} \rangle \rightarrow \text{id } \langle \text{arr} \rangle \mid \text{dt } \langle \text{arr} \rangle$

$\langle \text{arr} \rangle \rightarrow [\langle \text{arr}' \rangle]$

$\langle \text{arr}' \rangle \rightarrow ] \text{id } \langle \text{arr1} \rangle \mid , ] \text{id } \langle \text{arr1} \rangle$

$\langle \text{arr1} \rangle \rightarrow = \langle \text{arr1}' \rangle \mid \epsilon$

$\langle \text{arr1}' \rangle \rightarrow \text{new}[\langle \text{arr2} \rangle] \mid \{ \langle \text{values} \rangle \}$

$\langle \text{arr2} \rangle \rightarrow \text{dt}[\langle \text{oer} \rangle \langle \text{arr3} \rangle] \mid \text{id}[\langle \text{oer} \rangle \langle \text{arr3} \rangle]$

$\langle \text{arr3} \rangle \rightarrow ] \{ \langle \text{values} \rangle \} \mid , \langle \text{oer} \rangle ] \{ \langle \text{values} \rangle \}$

$\langle \text{values} \rangle \Rightarrow \langle \text{oer} \rangle \langle \text{v} \rangle$

$\langle \text{values}' \rangle \rightarrow \{ \langle \text{values} \rangle \} \langle \text{v1} \rangle$

$\langle \text{v} \rangle \rightarrow , \langle \text{oer} \rangle \langle \text{v} \rangle \mid \epsilon$

$\langle \text{v1} \rangle \rightarrow , \langle \text{values}' \rangle \mid \epsilon$

$\langle P \rangle \rightarrow dt \langle P_1 \rangle \mid id \langle P_1 \rangle \mid \epsilon$

$\langle P_1 \rangle \rightarrow id \langle P_1' \rangle$  ~~dt~~ ~~dt~~ ~~dt~~ ~~dt~~

$\langle P_2 \rangle \rightarrow ] \langle P_3 \rangle \mid , ] \langle P_3 \rangle$

$\langle P_3 \rangle \rightarrow , \langle P' \rangle \mid \epsilon$

$\langle P' \rangle \rightarrow id \langle P_1 \rangle \mid dt \langle p1 \rangle$

$\langle P_1' \rangle \rightarrow [ \langle P_2 \rangle \mid \epsilon$

$\langle type' \rangle \rightarrow [ \langle t_2 \rangle \mid \epsilon$

$\langle t_2 \rangle \rightarrow .] \mid , ]$

$\langle \text{dec} \rangle \rightarrow \text{dt id } \langle \text{init} \rangle \langle \text{list} \rangle \mid \text{id id } \langle \text{init} \rangle$   
 $\langle \text{list} \rangle$

$\langle \text{list} \rangle \rightarrow , \text{id } \langle \text{init} \rangle \langle \text{list} \rangle \mid \epsilon$

$\langle \text{init} \rangle \rightarrow = \langle \text{dec} \rangle \mid \epsilon$

$\langle \text{fun-def} \rangle \rightarrow \langle \text{static} \rangle \langle \text{type} \rangle \text{fun id}$   
 $(\langle \text{P} \rangle) \langle \text{ns} \rangle \{ \langle \text{h2MST} \rangle \}$

$\langle \text{static} \rangle \rightarrow \text{static } \mid \epsilon$

$\langle \text{type} \rangle \rightarrow \text{dt } \langle \text{type}' \rangle \mid \text{void} \mid \text{id } \langle \text{type}' \rangle$

$\langle \text{type}' \rangle \rightarrow [ \langle \text{t2} \rangle \mid \epsilon$

$\langle \text{t2} \rangle \rightarrow ] \mid , ]$

$\langle F' \rangle \rightarrow \langle F \rangle \mid \epsilon$

$\langle oe \rangle \rightarrow \langle ae \rangle \langle oe' \rangle$

$\langle oe' \rangle \rightarrow OR \langle ae \rangle \langle oe' \rangle \mid \epsilon$

$\langle ae \rangle \rightarrow \langle re \rangle \langle ae' \rangle$

$\langle ae' \rangle \rightarrow AND \langle re \rangle \langle ae' \rangle \mid \epsilon$

$\langle re \rangle \rightarrow \langle e \rangle \langle re' \rangle \mid \epsilon$

$\langle re' \rangle \rightarrow ROP \langle e \rangle \langle re' \rangle \mid \epsilon$

$\langle e \rangle \rightarrow \langle T \rangle \langle e' \rangle$

$\langle e' \rangle \rightarrow PM \langle T \rangle \langle e' \rangle \mid \epsilon$

$\langle T \rangle \rightarrow \langle F \rangle \langle T' \rangle$

$\langle T' \rangle \rightarrow MDM \langle F \rangle \langle T' \rangle \mid \epsilon$

$\langle F \rangle \rightarrow \langle \text{Const} \rangle \mid (\langle e \rangle) \mid !\langle F \rangle \mid \text{inc-dec} \langle x \rangle$   
 $\mid \text{id} \langle F' \rangle$

$\langle Z \rangle \rightarrow \cdot id \langle Z_2 \rangle \mid [ \langle o e \rangle \langle W_1 \rangle \langle Z_1 \rangle ] inc-dec$

$\downarrow$

$\langle Z_1 \rangle \rightarrow \cdot id \langle Z \rangle \mid inc-dec$

$\langle Z_2 \rangle \rightarrow id \langle Z \rangle \mid [ \langle o e \rangle \langle W_1 \rangle \langle Z_1 \rangle ] \epsilon$

<fun-calls> → id <y>

<Assign-st> → ~~id~~ <x> <Assign-op>  
<0e>

<inc-dec-st> → <x> incdec | inc-dec <x>

$\langle Y \rangle \rightarrow \cdot id \langle Y \rangle \backslash (\langle PL \rangle) \langle Y_1 \rangle \mid \langle ar \rangle . id \langle Y \rangle$

$\langle Y_1 \rangle \rightarrow \cdot id \langle Y \rangle \mid \langle ar \rangle . id \langle Y \rangle \in$

$\langle ar \rangle \rightarrow \langle \text{loop} \rangle \langle W1 \rangle$

$\langle x \rangle \rightarrow id \langle x_1 \rangle$

$\langle x_1 \rangle \rightarrow . id \langle x_2 \rangle \mid \epsilon \mid \langle x_3 \rangle$

$\langle x_2 \rangle \rightarrow \langle x_4 \rangle \mid \langle x_1 \rangle \mid \epsilon$

$\langle x_3 \rangle \rightarrow [ \langle \text{does} \langle w_1 \rangle \langle x_1 \rangle ]$

$\langle x_4 \rangle \rightarrow ( \langle RL \rangle ) \langle x_4 \rangle$

$\langle x_4' \rangle \rightarrow \langle x_3 \rangle \mid \langle x \rangle$

$\langle \text{SST} \rangle \rightarrow \langle \text{while-st} \rangle | \langle \text{do-while} \rangle | \langle \text{if-else} \rangle |$   
 $\langle \text{for-st} \rangle | \langle \text{return-st} \rangle | \langle \text{break-st} \rangle |$   
 $\langle \text{Continue-st} \rangle | \text{inc-dec } id \langle X \rangle |$   
 $id \langle \text{SST1} \rangle | dt \langle \text{SST2} \rangle$

id < SST1> | dt < SST2

void func\_id (<P>) <n> {<n><MS>  
 ^  
 }  
 4

$\langle SST1 \rangle \rightarrow \langle W \rangle$  |  $\Gamma \quad \langle SST1' \rangle$  |  $\vdash \langle SST3 \rangle | \langle SST5 \rangle$   
 $\langle SST1' \rangle \rightarrow$  ]  $\langle SST4 \rangle$  | ]  $\langle SST3 \rangle$

`<SSST2> → id <init> <list> | [ <SSST1> | <SSST2> ]`

$$\angle SST37 \rightarrow = \angle SST317$$

`<SSST3'> → new id <4> | 20e>`

`<SST4> → id<arr1> | &SST5>`

$\langle SST5 \rangle \rightarrow \text{func } \text{rd}(\langle p \rangle) \leftarrow \begin{cases} \langle h \rangle \langle MST \rangle \\ \dots \end{cases}$

$\langle w \rangle \rightarrow .id \langle w \rangle \mid [ \langle oe \rangle \langle w_1 \rangle \langle w_2 \rangle ]$

$\text{inc-dec} \mid \langle \text{assign-up} \rangle \langle oe \rangle$

$( \langle \text{PL} \rangle ) \quad \langle w_3 \rangle$

$\langle w_1 \rangle \rightarrow J \mid ; \langle oe \rangle ]$

$\langle w_2 \rangle \rightarrow .id \langle w \rangle \mid \text{inc-dec} \mid \langle \text{Assign\_op} \rangle \langle oe \rangle$

$\langle w_3 \rangle \rightarrow .id \langle w \rangle \mid [ \langle oe \rangle \langle w_1 \rangle \langle w_2 \rangle ]$

$\mid \epsilon$

$\langle \text{defs} \rangle \rightarrow \text{public } \langle \text{defs}' \rangle \quad | \quad \langle \text{defs}' \rangle \quad | \quad \epsilon$

$\langle \text{defs}' \rangle \rightarrow \langle \text{class\_def} \rangle \langle \text{rhs} \rangle \langle \text{defs}' \rangle$   
 $\langle \text{struct\_def} \rangle \langle \text{rhs} \rangle \langle \text{defs}' \rangle$

$\langle \text{struct-def} \rangle \rightarrow \text{struct } \text{id} \langle \text{hs} \rangle \{ \langle \text{hs} \rangle \langle \text{sbody} \rangle$

`<sbodys> → <act> <sbodys1>`

'body' → void . Func id (<P>) {<n> {<n> <NST>  
g | dt <sb> | id <sb1>

$\langle \text{SBS} \rangle \rightarrow \langle \text{type}'s \text{ Func id } (\langle P \rangle) \langle n \rangle \{ \langle n \rangle$   
 $\langle \text{MST} \rangle \} \mid \langle \text{arr} \rangle \mid \text{id} \langle \text{init} \rangle \langle \text{list} \rangle$

(sb1)  $\rightarrow$   $\langle \text{args} \rangle \parallel (\langle p \rangle \{ \langle n \rangle \langle \text{cons-body} \rangle \})$   
|  $\langle \text{type}' \rangle \text{ func id } (\langle p \rangle \langle n \rangle \{ \langle n \rangle \langle \text{MST} \rangle \})$

Less → public / private | E

$\langle \text{cons-body} \rangle \rightarrow \langle \text{Assign-st} \rangle \langle n \rangle \langle \text{cons-body} \rangle$  | ε

$\langle \text{class\_def} \rangle \rightarrow \langle \text{ab\_seal} \rangle \langle \text{static} \rangle \text{ class}$   
 $\quad \text{id} \langle \text{inh} \rangle \langle \text{ns} \rangle \{ \langle \text{ns} \rangle$   
 $\quad \quad \langle \text{body} \rangle \}$

$\langle \text{body} \rangle \rightarrow \langle \text{acc\_mod} \rangle \langle \text{static} \rangle \langle \text{body}' \rangle$   
 $\quad \langle \text{ns} \rangle \langle \text{body} \rangle \mid \epsilon$

$\langle \text{body}' \rangle \rightarrow \langle \text{ab\_seal} \rangle \langle \text{c} \rangle \mid \text{id} \langle \text{c1} \rangle \mid \text{dt} \langle \text{c2} \rangle$

$\langle \text{c} \rangle \rightarrow \text{dt} \langle \text{c}' \rangle \mid \text{id} \langle \text{c3} \rangle \mid \text{void, func id} (\langle \text{p} \rangle)$   
 $\quad \quad \quad \{ \langle \text{ns} \rangle \langle \text{MST} \rangle \}$

$\langle \text{c}' \rangle \rightarrow \text{id} \langle \text{init} \rangle \langle \text{list} \rangle \mid \langle \text{type}' \rangle \text{ func id} (\langle \text{p} \rangle) \{ \langle \text{ns} \rangle$   
 $\quad \quad \quad \langle \text{MST} \rangle \}$

$\langle \text{c1} \rangle \rightarrow \langle \text{arr} \rangle (\langle \text{p} \rangle) \{ \langle \text{ns} \rangle \langle \text{cons\_body} \rangle \}$   
 $\quad \quad \quad \mid \text{id} = \text{new id} \langle \text{y} \rangle$

$\langle \text{c2} \rangle \rightarrow \langle \text{arr} \rangle$

$\langle \text{c3} \rangle \rightarrow \langle \text{type}' \rangle \text{ func id} (\langle \text{p} \rangle) \{ \langle \text{ns} \rangle \langle \text{MST} \rangle \}$   
 $\quad \quad \quad \mid \text{id} \langle \text{init} \rangle \langle \text{list} \rangle$

$\langle \text{inh} \rangle \rightarrow : \text{id} \mid \epsilon$

$\langle \text{ab\_seal} \rangle \rightarrow \text{abstract} \mid \text{sealed} \mid \epsilon$

$\langle \text{static} \rangle \rightarrow \text{static} \mid \epsilon$

$\langle S \rangle \rightarrow \text{public class id } \langle \text{inh} \rangle \{ \langle h \rangle$

$\langle S1 \rangle \mid \text{class id } \langle \text{inh} \rangle \{ \langle h \rangle \langle S1 \rangle \}$

$\langle S1 \rangle \rightarrow \text{public } \langle S2 \rangle \mid \text{private } \langle \text{static} \rangle \langle \text{cbody}' \rangle$

$\mid \text{protected } \langle \text{static} \rangle \langle \text{cbody}' \rangle \langle h \rangle \langle \text{body} \rangle$

$\mid \langle \text{static} \rangle \langle \text{cbody}' \rangle \langle h \rangle \langle \text{cbody} \rangle$

$\langle S2 \rangle \rightarrow \text{static } \langle S3 \rangle \mid \langle \text{cbody}' \rangle \langle h \rangle \langle \text{body} \rangle$

$\langle S3 \rangle \rightarrow \text{void } \langle S4 \rangle \mid \text{dt } \langle C' \rangle \mid \text{id } \langle C \rangle$

$\mid \text{abstract } \langle S5 \rangle \mid \text{sealed } \langle S5 \rangle$

$\langle S4 \rangle \rightarrow \text{dt } \langle C' \rangle \mid \text{id } \langle C \rangle \mid \text{void func id}$

$(\langle P \rangle) \{ \langle h \rangle \langle M \rangle \}$

$\langle S5 \rangle \rightarrow \text{main } () \{ \langle h \rangle \langle M \rangle \} \langle h \rangle \langle \text{cbody} \rangle$

$\langle h \rangle \langle \text{defs} \rangle \mid \text{func } () \text{id } (\langle P \rangle) \{$

$\langle h \rangle \langle M \rangle \}$

$\langle \text{ab-seal} \rangle \rightarrow$  Abstract cat = "Abstract"  
cat  
 $\downarrow$  sealed cat = "Sealed"  
IE cat = "General"

$\langle \text{inh} \rangle \rightarrow$  : ID N = Token[Index].vp

T.lookup - MT (N, out cat)

if ( $T \neq \text{null}$ )  
{ "undeclared" | identified }

if ( $T == \text{"struct"}$ )

"Class can't inherit  
from struct".

if ( $T == \text{"class"}$  &&  
cat = "sealed")

{ "sealed class can't  
be inherited" }

Print N

$\langle \text{Decl} \rangle \rightarrow \text{DT} \underset{T}{\sim} \text{token}[i].vp \quad \text{ID} \underset{T}{\sim} \text{token}[i].vp$

if (insert-FT( $N, T$ ) == false)  
reddeclaration

$\langle \text{INIT} \rangle$

$\langle \text{LIST} \rangle$

$\langle \text{INIT} \rangle \rightarrow =$

$\langle \text{QEX} \rangle$

if (compatibility  
 $(T, T_1, "="))$

{ "Type match  
error" }

$\langle \text{LIST} \rangle \rightarrow ; \text{id } N \underset{T}{\sim} \text{token}[i].vp$

if (insert-FT( $N, T$ ) == false)  
{ "Reddeclaration" }

$\langle \text{INIT} \rangle$

$\langle \text{LIST} \rangle$

<return-st> → return <r1> Type

<r1> → <der Type> | ε

<Access-mod> → public AM = "public"

| private AM = "private"

| protected AM = "protected"

| ε AM = "private"

<Static> → Static TM = "static"

| ε

$\langle P \rangle \rightarrow dt \quad T = \text{tokenIndex}.vp \quad \langle P_1 \rangle$

$P_S$

$A, T$

$| \exists d \quad N = \text{tokenIndex}.vp, \bar{T} = \text{lookup-MT}$

$(N; \text{outcd}, \text{atrd})$

$\langle P_1 \rangle$

$\backslash \epsilon^{P_1}$

$\nabla \text{if } (T = \text{null})$

{ "undeclared" }

$\langle P_1 \rangle \rightarrow id \quad N = \text{tokenIndex}.vp$

$P, T$

$\text{if } (!\text{insert-FT}(N, T))$

{ "redeclaration" }

$P = T$

$\langle P_1' \rangle \quad \langle P_3 \rangle \quad P \neq "", "P_1"$

$\langle P_2 \rangle \rightarrow \exists \langle P_3 \rangle \quad |, ] \quad \langle P_3 \rangle$

$\langle P_3 \rangle \rightarrow , \quad \langle P' \rangle \quad \backslash \epsilon$

$\langle P' \rangle \rightarrow id \quad N = \text{tokenIndex}.vp, \bar{T} = \text{lookup-MT}$

$P, T$

$(N; \text{outcd}, \text{atrd})$

$\langle P_1 \rangle$

$\nabla \text{if } (T = \text{null})$

{ "undeclared" }

$| \quad dt \quad T = \text{tokenIndex}.vp \quad \langle P_1 \rangle$

$P, T$

AM = "public"

$\langle S \rangle \rightarrow \text{public class } T = \text{"class"}$

$i \in M \models \text{token}[index].vp$   $\langle \text{inhs} \rangle$   
 $\text{print}$

$CT\_ref \text{ create-DT}()$

$\text{if}(\text{insert-MT}(N; T, Cat, Print, Ref))$   
S

"Redeclaration" ?

{  $\text{create-Scope}()$   $\langle n \rangle \quad \langle S_1 \rangle$   
 $CT\_ref$

$\langle S_1 \rangle \rightarrow \text{public AM} = \text{token}[index].vp \quad \langle S_2 \rangle$   
 $CT\_ref, AM, C$

|  $\text{private AM} = \text{token}[index].vp \quad \langle \text{static} \rangle$   
 $T.M$

$\langle C\_body' \rangle \quad \langle n \rangle \quad \langle C\_body \rangle$   
 $AM, TM \quad CT\_ref \quad CT\_ref$

|  $\text{protected AM} = \text{token}[index].vp \quad \langle \text{static} \rangle$   
 $T.M$

$\langle C\_body' \rangle \quad \langle n \rangle \quad \langle C\_body \rangle$   
 $AM, TM, CT\_ref \quad CT\_ref$

|  $\langle \text{static} \rangle \quad \langle C\_body' \rangle \quad \langle n \rangle \quad \langle C\_body \rangle$   
 $TM \quad AM, TM, CT\_ref \quad CT\_ref$

$\langle S2 \rangle \rightarrow \text{static } TM = \text{"static"}$   $\langle S3 \rangle$   
 $AM, TM, CT-\text{ref}$

|  $\langle cbody' \rangle$  :  $\langle h \rangle \langle cbody \rangle$   
 $AM, TM, CT-\text{ref}$   $CT-\text{ref}$

$\langle S3 \rangle \rightarrow \text{void } T = \text{"void"}$   $\langle S4 \rangle$   
 $T, AM, TM, CT-\text{ref}$

|  $dt \quad T = \text{Token}[\text{index}].vp$   $\langle C' \rangle$   
 $T, AM, TM, CT-\text{ref}$

|  $\text{abstract cat} = \text{"abstract"}$   $\langle S5 \rangle$   
 $CAT, T, AM, TM, CT-\text{ref}$

|  $\text{id } N = \text{Token}[\text{index}].vp$   $\langle E3 \rangle$   
~~return~~  $TM, N, T, AM, CT-\text{ref}$

$T = \text{lookup\_MT}(N, \text{out cat})$

$\text{if } (T == \text{null})$

{ "undeclared" }

$\text{if } (T == \text{"struct"})$

{ "struct can't use" }

15

if (!insert-ST(N, T, AM, CT-ref))

{ "redeclaration" }

<S4> → main N : Token[index].vp ( Create-Scope )

) { <n> <MST> } scope-destroy( CT-ref )

<n> <Cbody> } scope-destroy( CT-ref )

| Func id N : Token[index].vp

( Create-Scope() <p> if (!insert-ST(N,  
P, "→", T, AM, CT-ref))  
CT-ref ))

{ "redeclaration" }

) { <n> <MST> } destroyScope( CT-ref )

<S5> → dt T : Token[index].vp <C'>

T, AM, TM, CT-ref

| void T = "void" Func id N : Token[index].vp

( Create-Scope() <p> if (!insert-ST(N, P, "→", T,  
AM, TM, CT-ref))

{ "redeclaration" }

) { <n> <MST> } destroyScope( CT-ref )

1 id N: tokent index].vp

<C3>

TM, N, T, AM, CT-~~ref~~

T = lookup-MT (N : out. cat)

if ( $T = \text{null}$ )  
{ "undeclared Class" }

if ( $T = \text{"struct"}$ )  
{

    { "struct" }  
    { "union" }  
    { "enum" }

if ( $T = \text{function}$ )  
    { "function" }

$\langle \text{OE} \rangle \rightarrow \langle \text{AE} \rangle$   $T_1$   $\langle \text{OE}' \rangle$   $T_1, \bar{T}$

$\langle \text{OE}' \rangle \rightarrow \text{OR}$   $T_1, \bar{T}$   $\langle \text{AE} \rangle$   $T_2$   $T_3 = \text{Compatibility}(T_1, T_2, \text{op})$

$\{ \text{if } (T_3 == \text{null}) \{ \text{"Type error"} \} \}$

$\langle \text{OE}' \rangle$   $T_3, \bar{T}$   $| \epsilon$

$\langle \text{AE} \rangle \rightarrow \langle \text{RE} \rangle$   $T_1$   $\langle \text{AE}' \rangle$   $T_1, \bar{T}$

$\langle \text{AE}' \rangle \rightarrow \text{AND}$   $T_1, \bar{T}$   $\text{op} = \text{token}[\text{index}] \cdot \text{vp}$   $\langle \text{RE} \rangle$   $T_2$

$T_3 = \text{Compatibility}(T_1, T_2, \text{op})$

$\{ \text{if } (T_3 == \text{null}) \{ \text{"Type error"} \} \}$

$\langle \text{AE}' \rangle$   $T_3, \bar{T}$   $| \epsilon$

$\langle \text{RE} \rangle \rightarrow \langle E \rangle$   $T_1$   $\langle \text{RE}' \rangle$   $T_1, \bar{T}$

$\langle \text{RE}' \rangle \rightarrow \text{ROP}$   $T_1, \bar{T}$   $\text{op} = \text{token}[\text{index}] \cdot \text{vp}$   $\langle E \rangle$   $T_2$

$T_3 = \text{Compatibility}(T_1, T_2, \text{op})$

$\{ \text{if } (T_3 == \text{null}) \{ \text{"Type error"} \} \}$

$\langle \text{RE}' \rangle$   $T_3, \bar{T}$   $| \epsilon$

$$\frac{\langle E \rangle}{T} \rightarrow \frac{\langle T \rangle}{T_1} \quad \frac{\langle E \rangle}{T_1, T}$$

$\langle E' \rangle \rightarrow PM \quad op = tokenIndex \} \langle T \rangle$

$$T_3 = \text{compatibility} (T_1, T_2, op)$$

$\{ \text{if } (\neg T_3 = \text{null})$

```
S("type encor")3
```

$\angle F' \rightarrow$   $\angle E$

$$\langle \bar{\gamma} \gamma \rightarrow \langle f \rangle \rangle_{\text{Tr}} = \langle \bar{\gamma}' \rangle_{\text{Tr}}$$

$\langle T^1 \rangle_{T_1, T} \rightarrow MDM \quad op = token(index).vp \langle F \rangle_{T_2}$

$$T_3 = \text{Compatibility}(T_1, T_2, op)$$

if ( $T_3 = \text{null}$ )

§ "type error"

$$\frac{<\tau'>}{\tau_{3,\tau}} \quad | \varepsilon$$

$\langle F \rangle \rightarrow \langle \text{const} \rangle$

$T_1, CT\text{-ref}$

$| (\langle e \rangle)$

$! op\_token[i]$

$\langle F \rangle$

$\neg \text{Compatibility}(T_1, CR)$   
 $\neg \text{Compatibility}(op, T_1)$

$iS(T = \text{null})$

$f \text{ "Type error"}$   
 $T = T_2$

$| \text{index } op\_token[\text{index}].vp$

$\langle XY \rangle$

$T_1, CT\text{-ref}$

$T_2 = \text{Compatibility}(op, T_1)$

$iS(T = \text{null})$

$f \text{ "Type error"}$

$\text{return } T = T_2$

$| id \quad N = token[\text{index}].vp$

$\langle F' \rangle$

$T, N, CT\text{-ref}$

$| \epsilon$

$\langle F' \rangle \rightarrow \langle Z \rangle$

$T, N, CT\text{-ref}$

$\langle X_1 \rangle \rightarrow \text{id} \ N = \text{token}[\text{index}] . \text{vp}$   $\langle X_1 \rangle$

$T, CT\_ref$

$N, CTref,$   
 $S, flags$

$\langle X_1 \rangle \rightarrow \cdot \text{id} \ N = \text{token}[\text{index}] . \text{vp}$   $\langle X_2 \rangle //$

$N, flags, CTref \downarrow$

$N_1, flags, CTref$

if ( $flags == 0$ )

{  $T = \text{lookup\_MT}(N)$  }

if ( $T == \text{null}$ )

{  $T = \text{lookup\_FT}(N)$  }

if ( $T == \text{null}$ )

{ "undeclared" }

}

else if ( $T == \text{"class"}$ )

{  $\text{isStatic} M = 1$  }

$CT\_ref = ref\_static$

else {

$T = \text{lookup\_all\_BT}(CTref, N)$

{ if ( $T == \text{null}$ ) { "Undeclared" }

if ( $AM == \text{private}$ ) { "Can't access" }

if ( $TM \neq \text{isStaticM}$ )

{ "Can't access" }

4

$\langle X_2 \rangle \rightarrow \langle X_4 \rangle$  |  $\epsilon$   
N.flag, CT-ref      P, N, flag, CT-ref  
 $\downarrow$   
 $\langle X_1 \rangle$  |  $\epsilon$   
N.flag, CT-ref

$\langle X_4 \rangle \rightarrow (\langle PL \rangle) : \langle X_4' \rangle$   
N.flag, CT-ref      P, N, CT-ref  
 $T = \text{lookup\_ENT-DT} (N, P, CT-ref)$   
if ( $T \neq \text{null}$ ) { "undeclared" }

$\langle X_4' \rangle \rightarrow \langle X_3 \rangle$  |  $\langle X_5 \rangle$   
N.flag, Geref      T, CT-ref  
  
if (isStatic  
!= static)  
|| (AM ==  
"Private")  
\\$ "Access"  
Can't  
Access

<Z> → • if (Flag == 0)  
 { T = lookup-MT(N)  
   if (T == null)  
     { T = lookup-FT(N)  
       if (T == null)  
         { "undeclared"  
           else if (T == "class")  
             { is-Static-M = "static"  
               CT-ref = ref  
             }  
           else { T = lookup-CH-DT  
                   (N, CT-ref)  
                   if (T == null)  
                     { "undeclared"  
                   else if (AM == "private")  
                     { "Can't access"  
                   else if (TM == "static")  
                     { "Can't access"  
                   }  
                   }  
                   }  
                   }  
                   }

| inc-dec

op = token[1] or type

T<sub>1</sub> = lookup-FT(N, op)

if (T<sub>1</sub> == null)

{ "undeclared" }

T<sub>2</sub> = compatibility (op, T<sub>1</sub>)

if (T<sub>2</sub> == null)

{ "type not matched" }

| (<PL>) <Z<sub>2</sub>>

p, N, CT-ref

flag, N, CT-ref

T<sub>2</sub> = lookup-fn-DT(N, p, CTref)

if (T<sub>2</sub> == null)

{ "undeclared" }

if (isStatic.M != "static")

{ "AM != "private" }

{ "Can't Access" }

<Z<sub>1</sub>> → if (flag == 0)

N, T, CT-ref, flag  
isStatic.M

{ T = lookup-FT(N) }

if (T == null)

{ T = lookup-FT(N) }

if (T == null)

{ "undeclared" }

else if (T == "class")

{ isStatic-M = "Static" }

{ T-ref = ref }

}

def { T = lookup-att-DT(N, T-ref)

if (T == null)

{ "undeclared" }

if (AM == "private")

{ "can't access" }

if (TM != "static")

{ "can't access" }

y

1. inc-dec

op = token[index].vp

$\langle \text{Fun-def} \rangle \rightarrow \langle \text{static} \rangle \langle \text{type} \rangle \text{ func}$

AM, CT-ref

TM

T

id N = token[index].vp

(create-Scope()) < P>

P

if (+ insert-DT(N, P, "→", T, AM, TM,  
CT-ref))

{ "function redeclaration" }

) <h> { <h> <MST>

CT-ref

} destroy-Scope()

$\langle \text{static} \rangle \rightarrow \text{static}$  TM = "static"

TM

| ε

TM = null

$\langle \text{type} \rangle \rightarrow \text{dt}$  T = token[index].vp <type'>

| void T = "void"

| id N = token[index].vp

T = lookup-MT(N)

if (T == null)

{ "undeclared" }

```
else if( T == "struct" ||  
        cat == "sealed")  
{  
    "access denied"\n}
```

`<defs> → public AM = "public" <defs'>`

`AM`

---

`| <defs'> | E`

`AM`

<defs> → <class-def> AM <n> <defs>

| <struct-def> <n> <defs>  
| AM .

$\langle \text{Assign-st} \rangle \rightarrow \langle X \rangle$        $\langle \text{Assignop} \rangle$   
 $T$                    $T_1, CT\text{-ref}$                    $op$

$\langle \text{oer} \rangle$   
 $T_2, CT\text{-ref}$

$T = \text{compatibility}(T_1, T_2, op)$

if ( $T == \text{null}$ )

{ "type not matched" }

$\langle \text{inc-dec-st} \rangle \rightarrow \langle X \rangle$        $\langle \text{inc-dec} \rangle$   
 $T$                    $T_1, CT\text{-ref}$                    $op$

$T = \text{compatibility}(T_1, op)$

if ( $T == \text{null}$ )

{ "type not matched" }

inc-dec       $\langle X \rangle$   
op                   $T_1, CT\text{-ref}$

$T = \text{compatibility}(T_1, op)$

if ( $T == \text{null}$ )

{ "type not matched" }

$\langle \text{fun-call} \rangle \rightarrow \text{id} \quad N = \text{token}[\text{index}].\text{vp}$   
 $T, CT-\text{ref}$

$\langle Y \rangle$   
 $N, T, CT-\text{ref}$   
Flag, is Static-M

$\langle \text{for-st} \rangle \rightarrow \text{for create-Scope}() \langle e_1 \rangle :$   
 $T$

$\langle e_2 \rangle : \underset{T_1}{\text{if}}(T, \# = \text{bool})$   
 $\{ \underset{T_2}{\text{"type error"}} \}$

$\langle e_3 \rangle \underset{T_2}{\langle \text{ns} \rangle \langle b-\text{def} \rangle \text{ scope def}}$   
 $\} (T)$

$\text{if } (T_1 = T_2)$   
 $\{ \text{"error" } \}$

$\langle e_1 \rangle \rightarrow \underset{T}{\langle \text{dec} \rangle} \quad | \quad \langle \text{Assign-st} \rangle$   
 $\epsilon$

$\langle e_2 \rangle \rightarrow \underset{T}{\langle \text{oe} \rangle T} \quad | \quad \epsilon$

$\langle e_3 \rangle \rightarrow \underset{T}{\text{id}} \quad N = \text{token}[\text{index}].\text{vp}$

$\langle e_3 \rangle \rightarrow \underset{T, T_1}{\text{if-lookup-FT}(N, T, S_{\text{def}})}$   
 $\{ \text{if}(T = \text{null}) \}$   
 $\text{if-undefined}$

inc-dec

OP = token[ index + 1 ].op

< X >

Tij CT-ref

T = compatibility (Tij op)

if ( $\neg T$  is null)

" type not matched "

< if-else st >  $\rightarrow$  if create-scope()

CT-ref

< body > : < n >

if ( $\neg T$  ) = bool

{ error }

< body >  $\hookrightarrow$  destroyScope()

< op-else >

< op-else >  $\rightarrow$  else createScope() < n >

CT-ref  
↓ E < body > destroyScope()

$\text{if}(T \neq \text{"bool"})$

↑ "error"

$\langle \text{while} \neq \text{S1} \rangle \rightarrow \text{while } \langle \text{do} \rangle : \langle h \rangle$

CT-ref

T

$\langle \text{createScope} \rangle \quad \langle \text{body} \rangle \quad \langle \text{destroyScope} \rangle$

CT+SC

destroy

Scope

$\langle \text{do\_while} \rangle \rightarrow \text{do } \langle n \rangle \{ \text{createScope}()$

CT-ref

$\langle h \rangle \langle \text{MST} \rangle \quad \gamma \langle \text{destroyScope} \rangle$

CT-ref

$\langle h \rangle \quad \text{while } \langle \text{do} \rangle : \langle h \rangle$

T

$\text{if}(T \neq \text{"bool"})$   
 $\{ \text{"error"} \}$

### Dates

Struct def  $\longrightarrow$  acts  
 $T, AM$

street Tm = "stand" id N=tʃ:i;ʃ- vp

~~shady~~ → ~~caes~~  
T, AM, N

statics TM = "static" esbody's  
ATA, TMA,  
CT-<sub>ref</sub>

$\langle \pi \rangle$   $\langle \text{body} \rangle$   
 $\text{AM}, \overline{\text{IM}}, \overline{\text{CT}} - \lambda_{\text{ref}}$

sbobj' → void T='void' func  
 id ( <P<sub>2</sub> )  
 N<sub>i</sub> = T[i].vp  
 create\_scope();  
 destroy\_scope();

dt {  
 | AIST, cT-ref  
 | create\_scope();  
 | dt  
 | T = t[i].vp  
 | cT-ref  
 | if (!insert-AT (N[i], T[i], A[i], cT-ref))  
 | if (T == null) → Sunderland  
 | if (T == "struct") → count access  
 | if (T == func) → func

cT-ref  
 destroy\_scope();

id N<sub>i</sub> = t[i].vp  
 <SB<sub>2</sub>>  
 cT-ref  
 if (!insert-AT (N[i], T[i], A[i], cT-ref))  
 → redeclaration

id N<sub>i</sub> = T[i].vp ( <P<sub>2</sub> )  
 &  
 create\_scope();  
 destroy\_scope();

<MST>  
 MST  
 scope ← }  
 destroy\_scope();

counts | id Nu=t[i]-1p      cont. a list

~~executive scope~~ Date: \_\_\_\_\_

shift → cons | (eB  
const-body) destroy } Netliving scope } create scope

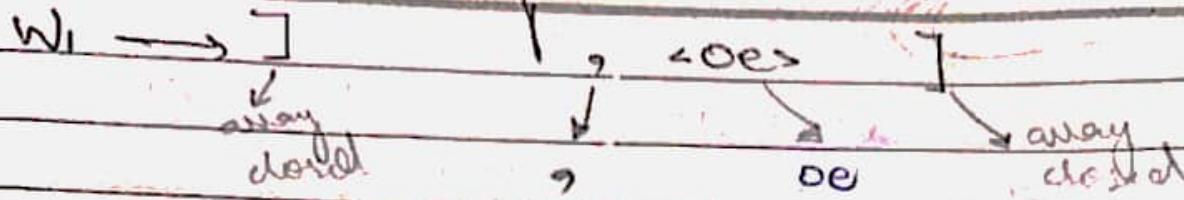
func id Netliving (eB  
create scope) } sheet scope

eMSTs MST } destroying scope

AM = "Public" / "Private"  
ac → Public | Private | E = "General"  
AM = t[ ] o VP

$W \rightarrow \cdot id \langle W_1 \rangle [coer] \langle W_1 \rangle \langle W_2 \rangle [inc-dec] \langle cast\_op \rangle [coer]$   
 $W_1 \rightarrow \langle PL \rangle \langle W_3 \rangle$   
 $W_2 \rightarrow \cdot id \langle W \rangle inc-dec \langle cast\_op \rangle [coer]$   
 $W_3 \rightarrow \cdot id \langle W_1 \rangle [coer] \langle W_1 \rangle \langle W_2 \rangle [inc-dec]$   
 $W \rightarrow \cdot id \ N = token \langle index \rangle \langle p \rangle \langle W \rangle$ 
Date: \_\_\_\_\_

if (flag == 0)  
 { T = lookup-NT(N)  
 if (T == null)  
 { T = lookup-RT(N)  
 if (T == null)  
 "undefined"  
 } else if (T == "class")  
 static M = 1  
 CT-ref = ref  
 else  
 T = lookup-RT(CT-S, N)  
 if (T == null) → "is declared"  
 if (NM = private → "can't access" (V))  
 if (TM != is static)  
 "can't access" } (CT) resolution



$W_2 \rightarrow id \sim W_2$

$$N = t[i].vp \quad w$$

line\\_dee  
 $N, T$   
 $inc ++$

assign\\_op coes

$W_3 \rightarrow id \quad N = t[i].vp \quad \text{coes} \quad | \quad \{ \text{coes } \sim W_1, \sim W_2 \}$

swipe  
vars

| E cat = "general"