P	::=	Program
	$\mid \overline{cld}$; s	Def. of a program.
cld	::=	Class definition
	class cn $\overline{fd}$ $\overline{method\_def}$	Def. of a class.
fd	::=	Field declaration
	cn f;	A field.
method_def	::=	Method declaration
	cn meth(cn var) {s; return x;}	Def. of methods
e	::=	Expression
	x	variable
	$null$	Null
	$e.f$	field access
	$e.meth(e)$	method invocation
	new $cn()$	object creation
	this	keyword
	le	labeled related expressions
le	::=	Label related expression
	l	label values
	labelData(e, l)	create a labeled data
	unlabel(e)	read label expression
	labelOf(e)	return the label of the expression
	unlabelOpaque(e)	read the opaquely labeled data
	opaqueCall(e.meth(e))	special method call
s	::=	Statement
	skip;	no-op
	x = e;	variable assignment
	x.f = e;	field write
	if *(x == y) s* ***else*** *s';*	condition branch
	e.m(e);	method call
	s s	sequence of statements

cn	Identifiers of class names
f	Identifiers of field names
method	Identifiers of method names
x, y, var	Identifiers of variables

Figure 1: Core Syntax of Java-like Programs

cl	::=	Expression Types
	| cn	A valid class name
	| **Label**	The class type for labels
	| $Labeled(cl)$	A type for labeled data
	| $OpaqueLabeled(cl)$	A type for opaquely labeled data
τ	::=	Statement Types
	| void	Void type
	| cl	Expression types
μ	::=	Method Types
	| $cl \rightarrow \tau$	

Figure 2: Types

config	::=	Configuration
	| (Σ, s)	(runtime env, statements)
Σ	::=	Runtime environment
	| (Δ, Ψ)	(Stack, Heap)
Δ	::=	Stack
	| []	empty stack
	| $\Delta = \Delta' \bullet \delta$	Stack is composed of frames
δ	::=	A labeled stack frame
	| (l, θ)	(label, a variable state)
θ	::=	Variable state
	| []	empty variable state
	| $\theta[x \mapsto v]$	θ with x mapping to v
Ψ	::=	Heap
	| []	Empty heap
	| $\Psi[\varsigma \mapsto \langle cn, \mathbb{F}, l \rangle]$	Object identifier ς maps to: (a) cn: Type of the object (b) $\mathbb{F}$: A mapping from fields to their values (c) l: Object id of the label for ς
$\mathbb{F}$	::=	Field state of object
	| []	empty field state
	| $\mathbb{F}[f \mapsto v]$	$\mathbb{F}$ with f mapping to v
v	::=	Values
	| ς	Object identifier
	| l	label variables
	| null	Null
	| NPE	Exception
Exception	::=	exception
	| NPE	Null pointer exceptions
Notations		
	$\Sigma.\Delta = \Delta$ and $\Sigma.\Psi = \Psi$	where Σ = (Δ, Ψ)
	$\delta.lbl = l$ and $\delta.\theta = \theta$	where δ = (l, θ)
	$\delta[x \mapsto v] = (l, \theta[x \mapsto v])$	where $\delta = (l, \theta)$
	$\Delta(x) = \theta(x)$	where $\Delta = \Delta' \bullet \delta$ and $\delta = (l, \theta)$
	$\Delta[x \mapsto v] = \Delta' \bullet \delta[x \mapsto v]$	where $\Delta = \Delta' \bullet \delta$
	$\Sigma.\Delta.lbl = l$	where $\Sigma.\Delta = \Delta' \bullet \delta$ and δ = (l, θ)
	$\Delta[lbl \mapsto l'] = \Delta' \bullet \delta'$	where $\Delta = \Delta' \bullet \delta$, δ = (l, θ) and $\delta' = (l', \theta)$

Figure 3: Abstract syntax representing program's state

Retrieve Method Body

$$findmbody(cl, meth) \stackrel{def}{=} \begin{cases} None & \text{method not found the in class cl} \\ (cl\text{" var, s; return y;}) & \text{where } md = cl\text{' meth(cl" var)\{ s; return y; \}} \end{cases}$$

where $cld = cl\ \overline{fd}\ \overline{method\_def}$, $md \in \overline{method\_def}$, and $cld \in$ P.

Retrieve Fields of a class

$$fields(cl) \stackrel{def}{=} \begin{cases} \emptyset & \text{No fields declared in the class definition} \\ (\overline{fd}) & \text{where } cld = cl\,\overline{fd}\ \overline{method\_def},\ \text{and } cld \in P. \end{cases}$$

Figure 4: Auxiliary functions

$$\textsc{Label}\ \ \Gamma \vdash l : \textbf{Label}$$

$$\textsc{Labeled}\ \ \frac{\Gamma \vdash e : cl \qquad \Gamma \vdash l : \textbf{Label}}{\Gamma \vdash labelData(e, l) : Labeled(cl)}$$

$$\textsc{RuntimeLabeled}\ \ \frac{\Gamma \vdash v : cl \qquad \Gamma \vdash l : \textbf{Label}}{\Gamma \vdash v^l : Labeled(cl)}$$

$$\textsc{Unlabel}\ \ \frac{\Gamma \vdash e : Labeled(cl)}{\Gamma \vdash unlabel(e) : cl}$$

$$\textsc{labelOf}\ \ \frac{\Gamma \vdash e : labeled(cl)}{\Gamma \vdash labelOf(e) : Label}$$

$$\textsc{unlabelOpaque}\ \ \frac{\Gamma \vdash e : OpaqueLabeled(cl)}{\Gamma \vdash unlabelOpaque(e) : cl}$$

$$\textsc{opaqueCall}\ \ \frac{\Gamma \vdash e : cl \qquad \Gamma \vdash l : \textbf{Label}}{\Gamma \vdash opaqueCall(e) : OpaqueLabeled(cl)}$$

Figure 5: Special Typing rules

E ::= [•] | *E s* | *return E;* | *E.f* | *E.f = e* | *x = E* | *E.meth(e)*

| *v.f = E* *(if E is not <unlabelOpaque(E)>)*

| *v.meth(E)* *(if E is not <unlabelOpaque(E)>)*

| *labelData(E, l)* | *unlabel(E)* | *labelOf(E)*

| *opaqueCall(E)* *(if E is not <return y;>)*

| *unlabelOpaque(E)*

Figure 6: Evaluation Context

E-VarRead

$$\frac{v = \Sigma.\Delta(x)}{\langle \Sigma, x \rangle \hookrightarrow \langle \Sigma, v \rangle}$$

E-FieldRead

$$\frac{\langle cn, \mathbb{F}, l \rangle = \Sigma.\Psi(\varsigma) \qquad v = \mathbb{F}(f) \qquad \varsigma = \Sigma.\Delta(x) \qquad l' = \Sigma.\Delta.lbl \sqcup l \qquad \Delta' = \Delta[lbl \mapsto l'] \qquad \Sigma' = \Sigma[\Delta \mapsto \Delta']}{\langle \Sigma, x.f \rangle \hookrightarrow \langle \Sigma', v \rangle}$$

E-MethodCall

$$\frac{\varsigma = \Sigma.\Delta(x) \qquad \langle cn, \mathbb{F}, l_x \rangle = \Sigma.\Psi(\varsigma) \qquad (\textit{cl var, s; return y;}) = findmbody(cn, m) \qquad l = \Sigma.\Delta.lbl \qquad l' = l_x \sqcup l \qquad \theta = \{this \mapsto \varsigma, var \mapsto v\} \qquad \delta = (l', \theta) \qquad \Delta' = \Sigma.\Delta \bullet \delta \qquad \Sigma' = \Sigma[\Delta \mapsto \Delta']}{\langle \Sigma, x.m(v) \rangle \hookrightarrow \langle \Sigma', \textit{s return y;} \rangle}$$

E-NewExpr

$$\frac{\varsigma \notin dom(\Sigma.\Psi) \qquad \mathbb{F} = \{f \mapsto null\}\ \forall f \in fields(cl) \qquad l = \Sigma.\Delta.lbl \qquad \Sigma' = \Sigma.\Psi[\varsigma \mapsto \langle cn, \mathbb{F}, l \rangle]]}{\langle \Sigma, \textit{new cn()} \rangle \hookrightarrow \langle \Sigma', \varsigma \rangle}$$

E-this

$$\frac{\varsigma = \Sigma.\Delta(this)}{\langle \Sigma, this \rangle \hookrightarrow \langle \Sigma, \varsigma \rangle}$$

Figure 7: Operational Semantics (part one)

E-labelData

$$\frac{}{\langle \Sigma, labelData(v, l) \rangle \hookrightarrow \langle \Sigma, v^l \rangle}$$

E-unlabel

$$\frac{l' = l \sqcup \Sigma.\Delta.lbl \qquad \Sigma' = \Sigma.\Delta[lbl \mapsto l']}{\langle \Sigma, unlabel(v^l) \rangle \hookrightarrow \langle \Sigma', v \rangle}$$

E-labelOf

$$\langle \Sigma, \textit{labelOf}(v^l) \rangle \hookrightarrow \langle \Sigma, l \rangle$$

E-callWithUnlablOpaque

$$\frac{\varsigma = \Sigma.\Delta(x) \qquad \langle cn, \mathbb{F}, l_x \rangle = \Sigma.\Psi(\varsigma) \qquad (\textit{cl var, s; return y;}) = \textit{findmbody(cn, m)} \qquad l' = l \sqcup \Sigma.\Delta.lbl \qquad \theta = \{this \mapsto \varsigma, var \mapsto v\} \qquad \delta = (l', \theta) \qquad \Delta' = \Sigma.\Delta \bullet \delta \qquad \Sigma' = \Sigma[\Delta \mapsto \Delta']}{\langle \Sigma, x.meth(unlabelOpaque(v^{[l]}) \rangle \hookrightarrow \langle \Sigma', \textit{s; return y;} \rangle}$$

E-opaqueCallReturn

$$\frac{\Sigma.\Delta = \Delta' \bullet \delta \qquad l = \Sigma.\Delta.lbl \qquad \Sigma' = \Sigma[\Delta \mapsto \Delta']}{\langle \Sigma, \textit{opaqueLabeled(return y;)} \rangle \hookrightarrow \langle \Sigma', v^{[l]} \rangle}$$

E-unlabelOpaque

$$\frac{l' = l \sqcup \Sigma.\Delta.lbl \qquad \Sigma' = \Sigma.\Delta[lbl \mapsto l']}{\langle \Sigma, unlabelOpaque(v^{[l]}) \rangle \hookrightarrow \langle \Sigma', v \rangle}$$

Figure 8: Operational Semantics for label expressions

S-VarAssign
$$\frac{x \in dom(\Sigma.\Delta) \qquad \Sigma' = \Sigma.\Delta[x \mapsto v]}{\langle \Sigma, x = v; \rangle \hookrightarrow \langle \Sigma', skip; \rangle}$$

S-FieldWrite
$$\frac{\varsigma = \Sigma.\Delta(x) \qquad \langle cn, \mathbb{F}, l \rangle = \Sigma.\Psi(\varsigma) \qquad \mathbb{F}' = \mathbb{F}[f = v] \qquad l' = \Sigma.\Delta.lbl \sqcup l \qquad \Sigma' = \Sigma.\Psi[\varsigma \mapsto \langle cl, \mathbb{F}', l' \rangle]}{\langle \Sigma, x.f = v; \rangle \hookrightarrow \langle \Sigma', skip; \rangle}$$

S-FieldWriteOpaque
$$\frac{\varsigma = \Sigma.\Delta(x) \qquad \langle cn, \mathbb{F}, l \rangle = \Sigma.\Psi(\varsigma) \qquad \mathbb{F}' = \mathbb{F}[f = v] \qquad l'' = \Sigma.\Delta.lbl \sqcup l' \qquad \Sigma' = \Sigma.\Psi[\varsigma \mapsto \langle cl, \mathbb{F}', l'' \rangle]}{\langle \Sigma, x.f = unlabelOpaque(v^{[l']}); \rangle \hookrightarrow \langle \Sigma', skip; \rangle}$$

S-Return
$$\frac{v = \Sigma.\Delta(x) \qquad \Sigma.\Delta = \Delta' \bullet \delta \qquad l = \delta.lbl \qquad \Delta'' = \Delta'[lbl \mapsto l \sqcup \Delta'.lbl] \qquad \Sigma' = \Sigma[\Delta \mapsto \Delta'']}{\langle \Sigma, \textit{return x;} \rangle \hookrightarrow \langle \Sigma', v \rangle}$$

S-if1
$$\frac{v_1 = v_2}{\langle \Sigma, \textbf{if}(v_1 == v_2) \textbf{ then } s \textbf{ else } s' \rangle \hookrightarrow \langle \Sigma, s \rangle}$$

S-if2
$$\frac{v_1 \neq v_2}{\langle \Sigma, \textbf{if}(v_1 == v_2) \textbf{ then } s \textbf{ else } s' \rangle \hookrightarrow \langle \Sigma, s' \rangle}$$

S-SeqV
$$\langle \Sigma, v; s \rangle \hookrightarrow \langle \Sigma, s \rangle$$

S-Seq
$$\langle \Sigma, \textit{skip; s} \rangle \hookrightarrow \langle \Sigma, s \rangle$$

Figure 9: Operational Semantics for statements