Name	a, b		
Class declaration	D	$::=$ `class` a `extends` s $\{\overline{d}\}$	
Super class	s	$::= a \mid$ `none`	
Member declaration	d	$::=$ `val` $a : T$	field declaration
		$\mid$ `def` $a(\overline{a} : \overline{T}) : T = t$	method definition
Term	t, u	$::= a$	variable, current instance
		$\mid$ `new` $a(\overline{t})$	instance creation
		$\mid t.a$	field selection
		$\mid t.a(\overline{t})$	method call
Type	T, U	$::= a$	class type
Program	P	$::= \overline{D}\ t$	
Class symbol	σ_c	$::=$ `Class`$(\overline{a}\mid\Gamma_f\mid\Gamma_m)$	($\overline{a}$ parents, Γ_f fields, Γ_m methods)
Field symbol	σ_f	$::=$ `Field`(T)	(T field type)
Method symbol	σ_m	$::=$ `Meth`$(\overline{T}\mid T)$	($\overline{T}$ parameter types, T result type)
Variable symbol	σ_v	$::=$ `Var`(T)	(T variable type)
Class scope	Γ_c	$::= \overline{a} \mapsto \overline{\sigma}_c$	
Field scope	Γ_f	$::= \overline{a} \mapsto \overline{\sigma}_f$	
Method scope	Γ_m	$::= \overline{a} \mapsto \overline{\sigma}_m$	
Variable scope	Γ_v	$::= \overline{a} \mapsto \overline{\sigma}_v$	

Notation	Interpretation	Condition
$\overline{a}$	sequence $a_1, \ldots, a_n$	$n \geq 0$
ϵ	empty sequence	
$\lvert\overline{a}\rvert$	length of sequence $\overline{a}$	
$\overline{a}, \overline{b}$	sequence concatenation	
$\overline{a} \mapsto \overline{\sigma}$	$a_1 \mapsto \sigma_1, \ldots, a_n \mapsto \sigma_n$	
$\mathrm{dom}(\overline{a} \mapsto \overline{\sigma})$	$\overline{a}$	
$\Gamma_c;\, \Gamma_v \vdash \overline{t} : \overline{T}$	$\Gamma_c;\, \Gamma_v \vdash t_1 : T_1, \ldots, \Gamma_c;\, \Gamma_v \vdash t_n : T_n$	$n = \lvert\overline{t}\rvert = \lvert\overline{T}\rvert$
$\Gamma_c \vdash \overline{D} \Rightarrow \Gamma'_c$	$\begin{cases} \Gamma_1 = \Gamma_c \\ \forall\, i \in [1, n],\ \Gamma_i \vdash D_i \Rightarrow \Gamma_{i+1} \end{cases}$	$n = \lvert\overline{D}\rvert,\ \Gamma'_c = \Gamma_{n+1}$
$\Gamma + (a \mapsto \sigma)$	$\Gamma, a \mapsto \sigma$	$a \notin \mathrm{dom}(\Gamma)$
$\Gamma + (a \mapsto \sigma)$	$\Gamma', a \mapsto \sigma, \Gamma''$	$\Gamma = \Gamma', a \mapsto \sigma', \Gamma''$
$\Gamma \uplus (a \mapsto \sigma)$	$\Gamma + (a \mapsto \sigma)$	$a \notin \mathrm{dom}(\Gamma)$
$\Gamma \uplus \{\epsilon\}$	Γ	
$\Gamma \uplus \{a \mapsto \sigma, \Gamma'\}$	$(\Gamma \uplus (a \mapsto \sigma)) \uplus \{\Gamma'\}$	

Fig. 1. Zwei Object-Oriented Fragment: Abstract Syntax, Symbols and Notations

Term typing

$(\Gamma_c; \Gamma_v \vdash t : T)$

(Ident)
$$\frac{a \mapsto \mathtt{Var}(T) \in \Gamma_v}{\Gamma_c; \Gamma_v \vdash a : T}$$

(Select)
$$\frac{\begin{array}{c}\Gamma_c; \Gamma_v \vdash t : b \\ b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m) \in \Gamma_c \\ a \mapsto \mathtt{Field}(T) \in \Gamma_f\end{array}}{\Gamma_c; \Gamma_v \vdash t.a : T}$$

(Call)
$$\frac{\begin{array}{c}\Gamma_c; \Gamma_v \vdash t : b \\ b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m) \in \Gamma_c \\ a \mapsto \mathtt{Meth}(\overline{T}|T) \in \Gamma_m \\ \Gamma_c; \Gamma_v \vdash \overline{t} : \overline{U} \quad \Gamma_c \vdash \overline{U} <: \overline{T}\end{array}}{\Gamma_c; \Gamma_v \vdash t.a(\overline{t}) : T}$$

(New)
$$\frac{\begin{array}{c}a \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m) \in \Gamma_c \\ \Gamma_f = \overline{a} \mapsto \mathtt{Field}(\overline{T}) \\ \Gamma_c; \Gamma_v \vdash \overline{t} : \overline{U} \quad \Gamma_c \vdash \overline{U} <: \overline{T}\end{array}}{\Gamma_c; \Gamma_v \vdash \mathtt{new}\ a(\overline{t}) : a}$$

Program typing

$(P \diamond)$

(Prog)
$$\frac{\begin{array}{c}\epsilon \vdash \overline{D} \Rightarrow \Gamma_c \\ \Gamma_c; \epsilon \vdash t : T\end{array}}{\overline{D}\ t \diamond}$$

Class typing

$(\Gamma_c \vdash D \Rightarrow \Gamma_c')$

(Class)
$$\frac{\begin{array}{c}\Gamma_c \vdash s \Rightarrow (\overline{a}|\Gamma_f|\Gamma_m) \\ \Gamma_c' = \Gamma_c \uplus (a \mapsto \mathtt{Class}(a, \overline{a}|\Gamma_f|\Gamma_m)) \\ \Gamma_c'; a \vdash \overline{d} \Rightarrow \Gamma_c''\end{array}}{\Gamma_c \vdash \mathtt{class}\ a\ \mathtt{extends}\ s\ \{\overline{d}\} \Rightarrow \Gamma_c''}$$

Superclass typing

$(\Gamma_c \vdash s \Rightarrow (\overline{a}|\Gamma_f|\Gamma_m))$

(None)
$$\frac{}{\Gamma \vdash \mathtt{none} \Rightarrow (\epsilon|\epsilon|\epsilon)}$$

(Super)
$$\frac{a \mapsto \mathtt{Class}(\overline{a}|\Gamma_f|\Gamma_m) \in \Gamma}{\Gamma \vdash a \Rightarrow (\overline{a}|\Gamma_f|\Gamma_m)}$$

Member typing

$(\Gamma_c; a \vdash d \Rightarrow \Gamma_c')$

(Field)
$$\frac{\begin{array}{c}\Gamma_c \vdash T \diamond \\ b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m) \in \Gamma_c \\ \Gamma_f' = \Gamma_f \uplus (a \mapsto \mathtt{Field}(T)) \\ \Gamma_c' = \Gamma_c + (b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f'|\Gamma_m))\end{array}}{\Gamma_c; b \vdash \mathtt{val}\ a : T \Rightarrow \Gamma_c'}$$

(Method)
$$\frac{\begin{array}{c}\Gamma_c \vdash T \diamond \quad \Gamma_c \vdash \overline{T} \diamond \\ b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m) \in \Gamma_c \\ a \mapsto \mathtt{Meth}(\overline{U}|U) \in \Gamma_m \text{ implies } \begin{cases}\Gamma_c \vdash \overline{U} <: \overline{T} \\ \Gamma_c \vdash T <: U\end{cases} \\ \Gamma_m' = \Gamma_m + (a \mapsto \mathtt{Meth}(\overline{T}|T)) \\ \Gamma_c' = \Gamma_c + (b \mapsto \mathtt{Class}(\overline{b}|\Gamma_f|\Gamma_m')) \\ \Gamma_v = \epsilon \uplus \{\text{this} \mapsto \mathtt{Var}(b), \overline{a} \mapsto \mathtt{Var}(\overline{T})\} \\ \Gamma_c'; \Gamma_v \vdash t : T' \quad \Gamma_c' \vdash T' <: T\end{array}}{\Gamma_c; b \vdash \mathtt{def}\ a(\overline{a} : \overline{T}) : T = t \Rightarrow \Gamma_c'}$$

Type typing

$(\Gamma_c \vdash T \diamond)$

(ClassType)
$$\frac{a \mapsto \mathtt{Class}(\overline{a}|\Gamma_f|\Gamma_m) \in \Gamma_c}{\Gamma_c \vdash a \diamond}$$

Subtyping

$(\Gamma_c \vdash T <: T')$

(SubClass)
$$\frac{a \mapsto \mathtt{Class}(\overline{a}|\Gamma_f|\Gamma_m) \in \Gamma_c}{\Gamma_c \vdash a <: a_i}$$

Fig. 2. OO Typing

Integer	n		
Unary operator	$unop$	$::= - \mid !$	
Binary operator	$binop$	$::= + \mid - \mid * \mid / \mid \% \mid ==$	
		$\mid\ != \mid < \mid \leq \mid > \mid \geq \mid \&\&$	
Term	t, u	$::= n$	integer literal
		$\mid\ \texttt{null}$	null reference
		$\mid\ unop\ t$	unary operation
		$\mid\ t\ binop\ t'$	binary operation
		$\mid\ \texttt{readInt}$	read integer
		$\mid\ \texttt{readChar}$	read character
		$\mid\ \texttt{if}\ (t)\ t'\ \texttt{else}\ t''$	conditional
		$\mid\ \{\overline{S}\ t\}$	block
		$\mid\ \ldots$	(as before)
Statement	S	$::= \texttt{while}\ (t)\ \{\overline{S}\}$	loop
		$\mid\ \texttt{var}\ a : T = t$	local variable
		$\mid\ \texttt{set}\ a = t$	variable assignment
		$\mid\ \texttt{do}\ t$	instruction
		$\mid\ \texttt{printInt}(t)$	print integer
		$\mid\ \texttt{printChar}(t)$	print character
Type	T	$::= \texttt{Int}$	integer type
		$\mid\ \texttt{Null}$	null type
		$\mid\ \ldots$	(as before)

Fig. 3. Additional Abstract Syntax

Term typing
$(\Gamma_c;\ \Gamma_v\ \vdash\ t : T)$

(IntLit)
$$\frac{}{\Gamma_c;\ \Gamma_v\ \vdash\ n : \texttt{Int}}$$

(NullLit)
$$\frac{}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{null} : \texttt{Null}}$$

(Unop)
$$\frac{\Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}}{\Gamma_c;\ \Gamma_v\ \vdash\ \mathit{unop}\ t : \texttt{Int}}$$

(Binop)
$$\frac{\begin{array}{c}\mathit{binop}\ \text{is not}\ \ ==\ \ \text{nor}\ !=\\ \Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}\\ \Gamma_c;\ \Gamma_v\ \vdash\ u : \texttt{Int}\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ t\ \mathit{binop}\ u : \texttt{Int}}$$

(ObjComp)
$$\frac{\begin{array}{c}\mathit{binop}\ \text{is}\ \ ==\ \ \text{or}\ !=\\ \Gamma_c;\ \Gamma_v\ \vdash\ t : T \qquad \Gamma_c;\ \Gamma_v\ \vdash\ u : U\\ \Gamma_c\ \vdash\ T <: U\ \text{or}\ \Gamma_c\ \vdash\ U <: T\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ t\ \mathit{binop}\ u : \texttt{Int}}$$

(ReadInt)
$$\frac{}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{readInt} : \texttt{Int}}$$

(ReadChar)
$$\frac{}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{readChar} : \texttt{Int}}$$

(If)
$$\frac{\begin{array}{c}\Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}\\ \Gamma_c;\ \Gamma_v\ \vdash\ t' : T \qquad \Gamma_c;\ \Gamma_v\ \vdash\ t'' : T'\\ \mathrm{lub}_{\Gamma_c}(T, T') = U\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{if}\ (t)\ t'\ \texttt{else}\ t'' : U}$$

(Block)
$$\frac{\begin{array}{c}\Gamma_c;\ \Gamma_v\ \vdash\ \overline{S}\ \Rightarrow\ \Gamma'_v\\ \Gamma_c;\ \Gamma'_v\ \vdash\ t : T\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ \{\overline{S}\ t\} : T}$$

Least upper bound
$(\mathrm{lub}_{\Gamma_c}(T, T') = U)$

(Lub)
$$\frac{\begin{array}{c}\Gamma_c\ \vdash\ T <: U\ \text{and}\ \Gamma_c\ \vdash\ T' <: U\\ \forall U'.\ \left.\begin{array}{c}\Gamma_c\ \vdash\ T <: U'\\ \Gamma_c\ \vdash\ T' <: U'\end{array}\right\}\ \Longrightarrow\ \Gamma_c\ \vdash\ U <: U'\end{array}}{\mathrm{lub}_{\Gamma_c}(T, T') = U}$$

Statement typing
$(\Gamma_c;\ \Gamma_v\ \vdash\ S\ \Rightarrow\ \Gamma'_v)$

(While)
$$\frac{\begin{array}{c}\Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}\\ \Gamma_c;\ \Gamma_v\ \vdash\ \overline{S}\ \Rightarrow\ \Gamma'_v\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{while}\ (t)\ \{\overline{S}\}\ \Rightarrow\ \Gamma_v}$$

(Var)
$$\frac{\begin{array}{c}\Gamma_c\ \vdash\ T\ \diamond\\ \Gamma_c;\ \Gamma_v\ \vdash\ t : U \qquad \Gamma_c\ \vdash\ U <: T\\ \Gamma'_v = \Gamma_v \uplus (a \mapsto \texttt{Var}(T))\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{var}\ a : T = t\ \Rightarrow\ \Gamma'_v}$$

(Set)
$$\frac{\begin{array}{c}a \mapsto \texttt{Var}(T) \in \Gamma_v\\ \Gamma_c;\ \Gamma_v\ \vdash\ t : U\\ \Gamma_c\ \vdash\ U <: T\end{array}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{set}\ a = t\ \Rightarrow\ \Gamma_v}$$

(Do)
$$\frac{\Gamma_c;\ \Gamma_v\ \vdash\ t : T}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{do}\ t\ \Rightarrow\ \Gamma_v}$$

(PrintInt)
$$\frac{\Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{printInt}(t)\ \Rightarrow\ \Gamma_v}$$

(PrintChar)
$$\frac{\Gamma_c;\ \Gamma_v\ \vdash\ t : \texttt{Int}}{\Gamma_c;\ \Gamma_v\ \vdash\ \texttt{printChar}(t)\ \Rightarrow\ \Gamma_v}$$

Type typing
$(\Gamma_c\ \vdash\ T\ \diamond)$

(IntType)
$$\frac{}{\Gamma_c\ \vdash\ \texttt{Int}\ \diamond}$$

(NullType)
$$\frac{}{\Gamma_c\ \vdash\ \texttt{Null}\ \diamond}$$

Subtyping
$(\Gamma_c\ \vdash\ T <: T')$

(IntRefl)
$$\frac{}{\Gamma_c\ \vdash\ \texttt{Int} <: \texttt{Int}}$$

(NullRefl)
$$\frac{}{\Gamma_c\ \vdash\ \texttt{Null} <: \texttt{Null}}$$

(SubNull)
$$\frac{}{\Gamma_c\ \vdash\ \texttt{Null} <: a}$$

Fig. 4. Typing of Additional Syntax