CS 4110

Programming Languages & Logics

Lecture 29
Featherweight Java

9 November 2016

CS 4110

Programming Languages & Logics

Lecture 29
Featherweight Java

9 November 2016

Announcements

- Homework #8 due tonight at 11:59pm
- No new homework out: Prelim II is next week

Roadmap

We've been building up from the λ-calculus to get languages resembling "real" functional programming languages like ML.

Can we use the same tools to formalize a very different kind of language?

Object-Oriented Features

Today we'll study a core calculus called *Featherweight Java*, developed by Igarashi, Pierce, and Wadler in 2002.

Featherweight Java : Java

::

polymorphic λ-calculus with references : OCaml

Object-Oriented Features

Today we'll study a core calculus called *Featherweight Java*, developed by Igarashi, Pierce, and Wadler in 2002.

Featherweight Java : Java
::
polymorphic λ-calculus with references : OCaml

Featherweight Java is small: it just has classes, inheritance, constructors, fields, methods, and casts, and it omits everything else.

Its simplicity makes its type soundness proof short and easy to extend.

Question

- MESSAGE PASSING
- METHOD CALLS

What *is* Object-Oriented Programming?

- ENCAPSULATION
- DYNAMIC DISPATCH

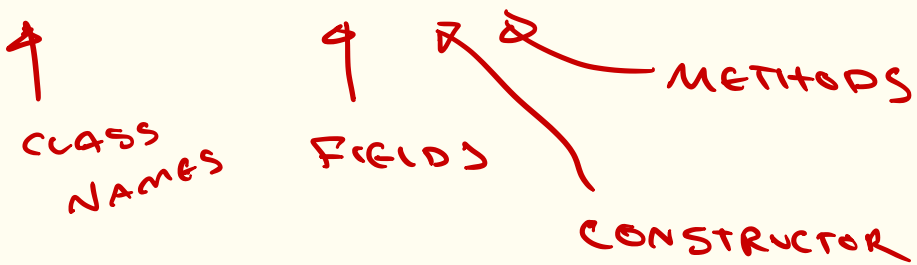
Syntax

$P \quad ::= \quad \overline{CL}\ e$ *programs*

$CL_0\ CL_1\ CL_2\ \cdots\ CL_n$

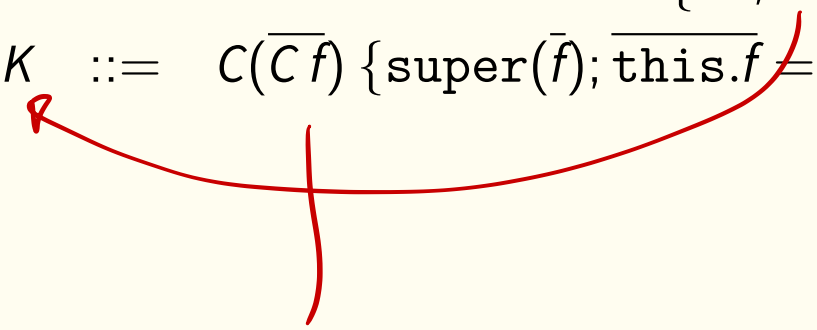
Syntax

$$
\begin{array}{rcll}
P & ::= & \overline{CL}\ e & \textit{programs} \\
CL & ::= & \texttt{class}\ C\ \texttt{extends}\ C\ \{\overline{C\ f};\ K\ \overline{M}\} & \textit{classes}
\end{array}
$$



Syntax

P	$::=$	$\overline{CL}\ e$	*programs*
CL	$::=$	$\texttt{class}\ C\ \texttt{extends}\ C\ \{\overline{C\ f};\ K\ \overline{M}\}$	*classes*
K	$::=$	$C(\overline{C\ f})\ \{\texttt{super}(\bar{f}); \overline{\texttt{this}.f = \bar{f};}\}$	*constructors*



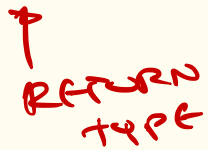
$C_0\ f_0,\ C_1\ f_1$

this.foo = foo

this.bar = bar

Syntax

P	$::=$	$\overline{CL}\ e$	*programs*
CL	$::=$	$\texttt{class}\ C\ \texttt{extends}\ C\ \{\overline{C\ f};\ K\ \overline{M}\}$	*classes*
K	$::=$	$C(\overline{C\ f})\ \{\texttt{super}(\bar{f}); \overline{\texttt{this}.f = \bar{f};}\}$	*constructors*
M	$::=$	$C\ m(\overline{C\ x})\{\texttt{return}\ e\}$	*methods*



Syntax

P	$::=$	$\overline{CL}\, e$	*programs*
CL	$::=$	$\texttt{class}\, C\, \texttt{extends}\, C\, \{\overline{C\, f};\ K\, \overline{M}\}$	*classes*
K	$::=$	$C(\overline{C\, f})\, \{\texttt{super}(\bar{f}); \overline{\texttt{this}.f = \bar{f};}\}$	*constructors*
M	$::=$	$C\, m(\overline{C\, x})\{\texttt{return}\, e\}$	*methods*
e	$::=$	x	*expressions*
	$\mid$	$e.f$	
	$\mid$	$e.m(\bar{e})$	
	$\mid$	$\texttt{new}\, C(\bar{e})$	
	$\mid$	$(C)\, e$	

CAST

class A ex. Obj {
  I field;
  A (field) {
    this. field;
  }
}

class B ext A {
  I field2;
  B (field, field2) {
    super (field);
  }
}

Syntax

P	$::=$	$\overline{CL}\ e$	*programs*
CL	$::=$	$\texttt{class}\ C\ \texttt{extends}\ C\ \{\overline{C\ f};\ K\ \overline{M}\}$	*classes*
K	$::=$	$C(\overline{C\ f})\ \{\texttt{super}(\bar{f}); \overline{\texttt{this}.f = \bar{f};}\}$	*constructors*
M	$::=$	$C\ m(\overline{C\ x})\{\texttt{return}\ e\}$	*methods*
e	$::=$	x	*expressions*
	$\mid$	$e.f$	
	$\mid$	$e.m(\overline{e})$	
	$\mid$	$\texttt{new}\ C(\overline{e})$	
	$\mid$	$(C)\ e$	
v	$::=$	$\texttt{new}\ C(\overline{v})$	*values*

$v ::= \lambda x.\ e$

Syntax

P	$::=$	$\overline{CL}\ e$	*programs*
CL	$::=$	$\texttt{class}\ C\ \texttt{extends}\ C\ \{\overline{C\ f};\ K\ \overline{M}\}$	*classes*
K	$::=$	$C(\overline{C\ f})\ \{\texttt{super}(\bar{f}); \overline{\texttt{this}.f = \bar{f};}\}$	*constructors*
M	$::=$	$C\ m(\overline{C\ x})\{\texttt{return}\ e\}$	*methods*
e	$::=$	x	*expressions*
	$\mid$	$e.f$	
	$\mid$	$e.m(\overline{e})$	
	$\mid$	$\texttt{new}\ C(\overline{e})$	
	$\mid$	$(C)\ e$	
v	$::=$	$\texttt{new}\ C(\overline{v})$	*values*
E	$::=$	$[\cdot]$	*evaluation contexts*
	$\mid$	$E.f$	
	$\mid$	$E.m(\overline{e})$	
	$\mid$	$v.m(\overline{v}, E, \overline{e})$	
	$\mid$	$\texttt{new}\ C(\overline{v}, E, \overline{e})$	
	$\mid$	$(C)\ E$	

Example

```
class A extends Object { A() { super(); } }
class B extends Object { A() { super(); } }
```

Example

```
class A extends Object { A() { super(); } }
class B extends Object { A() { super(); } }
 class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Object fst, Object snd) {
    super();
    this.fst = fst;
    this.snd = snd;
  }
  Pair swap() {
    return new Pair(this.snd, this.fst);
  }
}
```

Example

```
class A extends Object { A() { super(); } }
class B extends Object { A() { super(); } }
 class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Object fst, Object snd) {
    super();
    this.fst = fst;
    this.snd = snd;
  }
  Pair swap() {
    return new Pair(this.snd, this.fst);
  }
}
new Pair(new A(), new B()).swap()
```

class Main { static void main() { } }

Subtyping

$$\frac{}{C \leq C}\ \text{S-Refl}$$

Subtyping

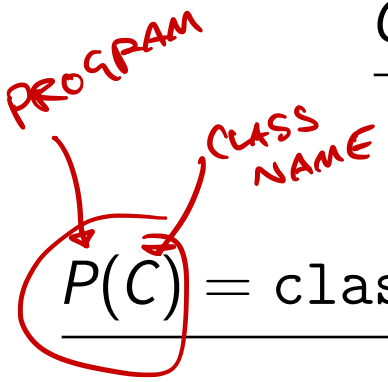
$$\frac{}{C \leq C}\ \text{S-Refl}$$

$$\frac{C \leq D \quad D \leq E}{C \leq E}\ \text{S-Trans}$$

Subtyping

$$\frac{}{C \leq C}\ \text{S-Refl}$$

$$\frac{C \leq D \quad D \leq E}{C \leq E}\ \text{S-Trans}$$



$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\}}{C \leq D}\ \text{S-Class}$$

Field Lookup

$fields(C) = \overline{F}$

$$\frac{}{\mathit{fields}(\texttt{Object}) = []}\ \textsc{F-Object}$$

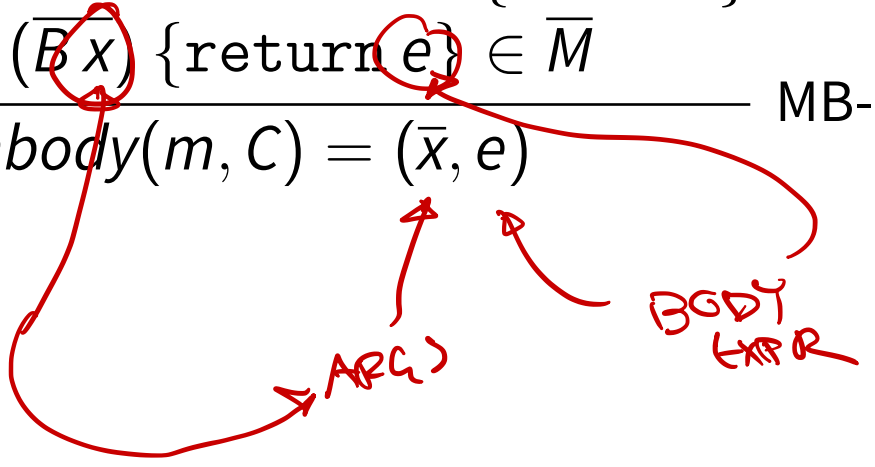
Field Lookup

$$\frac{}{\mathit{fields}(\texttt{Object}) = []}\ \textsc{F-Object}$$

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad \mathit{fields}(D) = \overline{D\ g}}{\mathit{fields}(C) = \overline{D\ g}\ @\ \overline{C\ f}}\ \textsc{F-Class}$$

Method Body Lookup

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \in \overline{M}}{mbody(m, C) = (\overline{x}, e)}\ \text{MB-Class}$$



Method Body Lookup

$B \quad B_0 \; B_1 \cdots B_n \quad mbody(swap, Pair) = (\overline{x}, e)$

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \in \overline{M}}{mbody(m, C) = (\overline{x}, e)}\ \text{MB-Class}$$

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \notin \overline{M}}{mbody(m, C) = mbody(m, D)}\ \text{MB-Super}$$

Operational Semantics

$$E ::= [\cdot] \mid E.f \mid E.m(\overline{e}) \mid v.m(\overline{v}, E, \overline{e}) \mid \texttt{new}\ C(\overline{v}, E, \overline{e}) \mid (C)\,E$$

Operational Semantics

$$E ::= [\cdot] \mid E.f \mid E.m(\overline{e}) \mid v.m(\overline{v}, E, \overline{e}) \mid \texttt{new}\, C(\overline{v}, E, \overline{e}) \mid (C)\, E$$

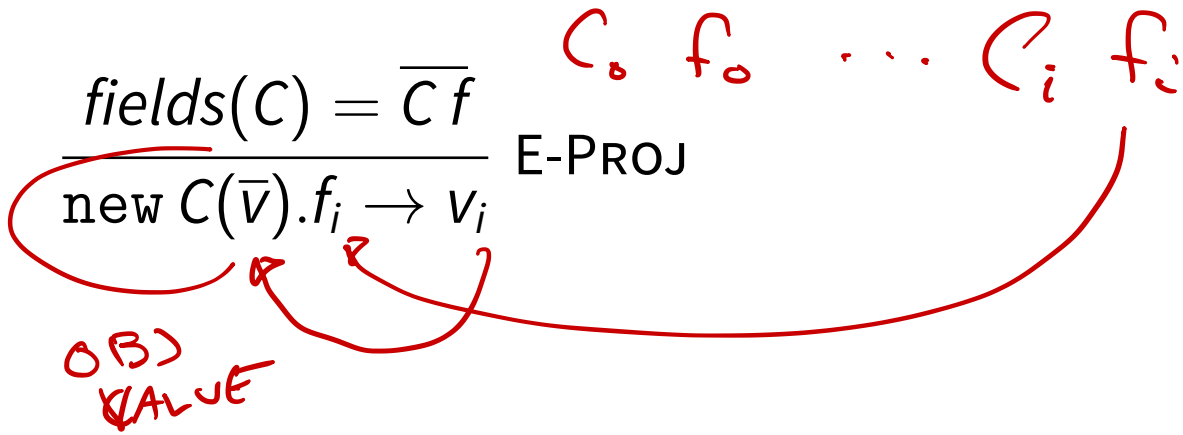
$$\frac{e \to e'}{E[e] \to E[e']}\ \textsc{E-Context}$$

Operational Semantics

$$E ::= [\cdot] \mid E.f \mid E.m(\overline{e}) \mid v.m(\overline{v}, E, \overline{e}) \mid \texttt{new}\, C(\overline{v}, E, \overline{e}) \mid (C)\, E$$

$$\frac{e \to e'}{E[e] \to E[e']}\ \textsc{E-Context}$$

$$\frac{\mathit{fields}(C) = \overline{C\ f}}{\texttt{new}\, C(\overline{v}).f_i \to v_i}\ \textsc{E-Proj}$$



Operational Semantics

$$E ::= [\cdot] \mid E.f \mid E.m(\overline{e}) \mid v.m(\overline{v}, E, \overline{e}) \mid \texttt{new}\, C(\overline{v}, E, \overline{e}) \mid (C)\, E$$

$$\frac{e \to e'}{E[e] \to E[e']}\ \text{E-Context}$$

$$\frac{\mathit{fields}(C) = \overline{C\, f}}{\texttt{new}\, C(\overline{v}).f_i \to v_i}\ \text{E-Proj}$$

$$\frac{\mathit{mbody}(m, C) = (\overline{x}, e)}{\texttt{new}\, C(\overline{v}).m(\overline{u}) \to [\overline{x} \mapsto \overline{u}, \texttt{this} \mapsto \texttt{new}\, C(\overline{v})]e}\ \text{E-Invk}$$

$e\{u_0/x_0\}\{u_1/x_1\} \cdots \{\texttt{new}\ C(\overline{v})/\texttt{this}\}$

Operational Semantics

$$E ::= [\cdot] \mid E.f \mid E.m(\overline{e}) \mid v.m(\overline{v}, E, \overline{e}) \mid \texttt{new}\, C(\overline{v}, E, \overline{e}) \mid (C)\, E$$

$$\frac{e \to e'}{E[e] \to E[e']}\ \textsc{E-Context}$$

$$\frac{\mathit{fields}(C) = \overline{C\, f}}{\texttt{new}\, C(\overline{v}).f_i \to v_i}\ \textsc{E-Proj}$$

$$\frac{\mathit{mbody}(m, C) = (\overline{x}, e)}{\texttt{new}\, C(\overline{v}).m(\overline{u}) \to [\overline{x} \mapsto \overline{u}, \texttt{this} \mapsto \texttt{new}\, C(\overline{v})]e}\ \textsc{E-Invk}$$

$$\frac{C \leq D}{(D)\, \texttt{new}\, C(\overline{v}) \to \texttt{new}\, C(\overline{v})}\ \textsc{E-Cast}$$

Method Type Lookup

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \in \overline{M}}{mtype(m, C) = \overline{B} \to B}\ \text{MT-Class}$$

Method Type Lookup

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \in \overline{M}}{mtype(m, C) = \overline{B} \to B}\ \text{MT-Class}$$

$$\frac{P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad B\ m\ (\overline{B\ x})\ \{\texttt{return}\ e\} \notin \overline{M}}{mtype(m, C) = mtype(m, D)}\ \text{MT-Super}$$

Typing Rules

$\Gamma \vdash e : \tau \; C$

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C} \text{ T-VAR}$$

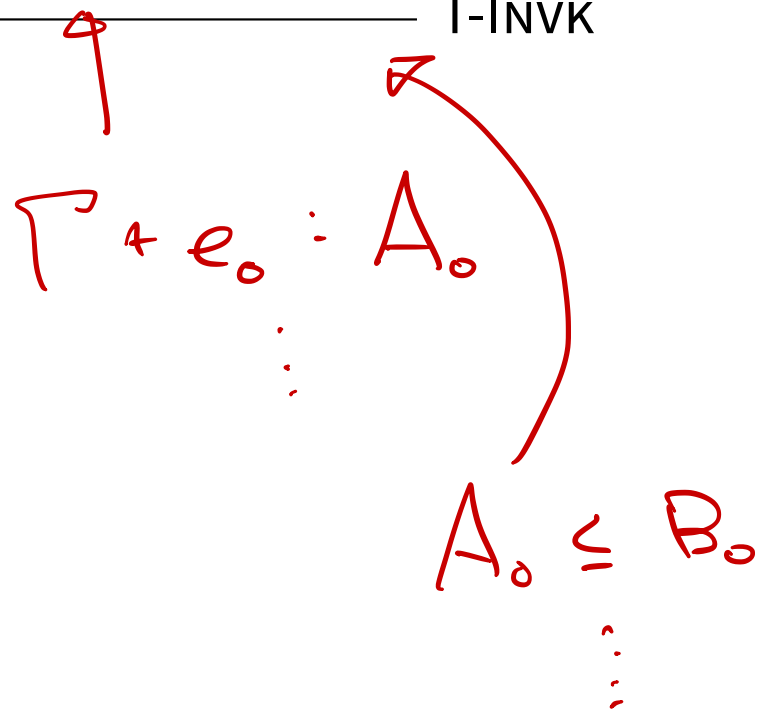
Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \textsc{T-Var} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \textsc{T-Field}$$

Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \textsc{T-Var} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \textsc{T-Field}$$

$$\frac{\Gamma \vdash e : C \qquad \mathit{mtype}(m, C) = \overline{B} \to B \qquad \Gamma \vdash \overline{e} : \overline{A} \qquad \overline{A} \leq \overline{B}}{\Gamma \vdash e.m(\overline{e}) : B}\ \textsc{T-Invk}$$



Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \textsc{T-Var} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \textsc{T-Field}$$

$$\frac{\Gamma \vdash e : C \qquad \mathit{mtype}(m, C) = \overline{B} \to B \qquad \Gamma \vdash \overline{e} : \overline{A} \qquad \overline{A} \leq \overline{B}}{\Gamma \vdash e.m(\overline{e}) : B}\ \textsc{T-Invk}$$

$$\frac{\mathit{fields}(C) = \overline{C\ f} \qquad \Gamma \vdash \overline{e} : \overline{B} \qquad \overline{B} \leq \overline{C}}{\Gamma \vdash \texttt{new}\ C(\overline{e}) : C}\ \textsc{T-New}$$

Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \text{T-VAR} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \text{T-FIELD}$$

$$\frac{\Gamma \vdash e : C \qquad \mathit{mtype}(m, C) = \overline{B} \to B \qquad \Gamma \vdash \overline{e} : \overline{A} \qquad \overline{A} \leq \overline{B}}{\Gamma \vdash e.m(\overline{e}) : B}\ \text{T-INVK}$$

$$\frac{\mathit{fields}(C) = \overline{C\ f} \qquad \Gamma \vdash \overline{e} : \overline{B} \qquad \overline{B} \leq \overline{C}}{\Gamma \vdash \texttt{new}\ C(\overline{e}) : C}\ \text{T-NEW}$$

$$\frac{\Gamma \vdash e : D \qquad D \leq C}{\Gamma \vdash (C)\, e : C}\ \text{T-UCAST}$$

Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \text{T-Var} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \text{T-Field}$$

$$\frac{\Gamma \vdash e : C \qquad \mathit{mtype}(m, C) = \overline{B} \to B \qquad \Gamma \vdash \overline{e} : \overline{A} \qquad \overline{A} \leq \overline{B}}{\Gamma \vdash e.m(\overline{e}) : B}\ \text{T-Invk}$$

$$\frac{\mathit{fields}(C) = \overline{C\ f} \qquad \Gamma \vdash \overline{e} : \overline{B} \qquad \overline{B} \leq \overline{C}}{\Gamma \vdash \texttt{new}\ C(\overline{e}) : C}\ \text{T-New}$$

$$\frac{\Gamma \vdash e : D \qquad D \leq C}{\Gamma \vdash (C)\, e : C}\ \text{T-UCast} \qquad \frac{\Gamma \vdash e : D \qquad C \leq D \qquad C \neq D}{\Gamma \vdash (C)\, e : C}\ \text{T-DCast}$$

Typing Rules

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C}\ \text{T-Var} \qquad \frac{\Gamma \vdash e : C \qquad \mathit{fields}(C) = \overline{C\ f}}{\Gamma \vdash e.f_i : C_i}\ \text{T-Field}$$

$$\frac{\Gamma \vdash e : C \qquad \mathit{mtype}(m, C) = \overline{B} \to B \qquad \Gamma \vdash \overline{e} : \overline{A} \qquad \overline{A} \leq \overline{B}}{\Gamma \vdash e.m(\overline{e}) : B}\ \text{T-Invk}$$

$$\frac{\mathit{fields}(C) = \overline{C\ f} \qquad \Gamma \vdash \overline{e} : \overline{B} \qquad \overline{B} \leq \overline{C}}{\Gamma \vdash \texttt{new}\ C(\overline{e}) : C}\ \text{T-New}$$

$$\frac{\Gamma \vdash e : D \qquad D \leq C}{\Gamma \vdash (C)\, e : C}\ \text{T-UCast} \qquad \frac{\Gamma \vdash e : D \qquad C \leq D \qquad C \neq D}{\Gamma \vdash (C)\, e : C}\ \text{T-DCast}$$

$$\frac{\Gamma \vdash e : D \qquad C \not\leq D \qquad D \not\leq C \qquad \textit{stupid warning}}{\Gamma \vdash (C)\, e : C}\ \text{T-SCast}$$

Method Typing

$$\frac{mtype(m, D) = \overline{A} \to A \text{ implies } \overline{A} = \overline{B} \text{ and } A = B}{override(m, D, \overline{B} \to B)} \textsc{Override}$$

Method Typing

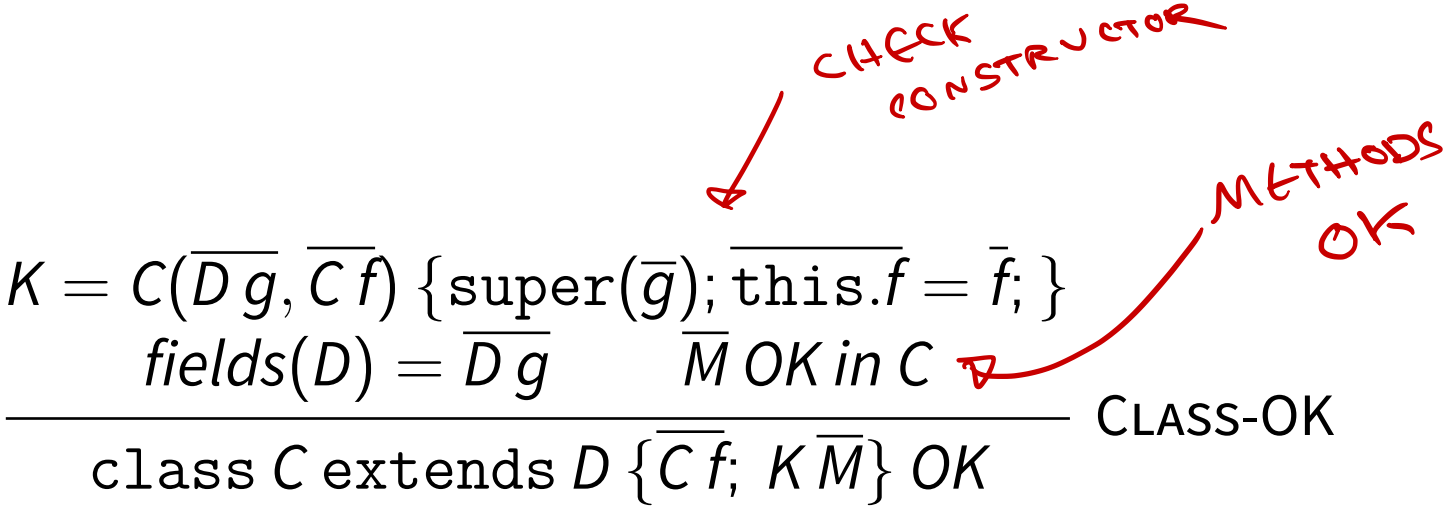
$$\frac{mtype(m, D) = \overline{A} \to A \text{ implies } \overline{A} = \overline{B} \text{ and } A = B}{override(m, D, \overline{B} \to B)} \textsc{Override}$$

ARGUMENTS

RETURN TYPE

$$\frac{\overline{x : B}, \texttt{this} : C \vdash e : A \qquad A \leq B \qquad P(C) = \texttt{class}\ C\ \texttt{extends}\ D\ \{\overline{C\ f};\ K\ \overline{M}\} \qquad override(m, D, \overline{B} \to B)}{B\ m(\overline{B\ x})\{\texttt{return}\ e\}\ OK\ in\ C} \textsc{Method-OK}$$

Class Typing



Type Soundness

We can prove type soundness in *almost* the standard way...

Type Soundness

We can prove type soundness in *almost* the standard way...

Lemma (Preservation)

If $\Gamma \vdash e : C$ *and* $e \to e'$ *then there exists a type* C' *such that* $\Gamma \vdash e' : C'$ *and* $C' \leq C$.

Type Soundness

We can prove type soundness in *almost* the standard way...

Lemma (Preservation)

If $\Gamma \vdash e : C$ *and* $e \to e'$ *then there exists a type* C' *such that* $\Gamma \vdash e' : C'$ *and* $C' \leq C$.

Lemma (Progress)

Let e be an expression such that $\vdash e : C$. *Then either:*

1. *e is a value,*
2. *there exists an expression* e' *such that* $e \to e'$, *or*
3. $e = E[(B)\,(\mathit{new}\, A(\overline{v}))]$ *with* $A \not\leq B$.