

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  $\lambda$ -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  $\lambda$ -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  $\lambda$ -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

---

What *is* Object-Oriented Programming?

# Syntax

$P ::= \overline{CL}e$

*programs*

# Syntax

$P ::= \overline{CL} e$  *programs*  
 $CL ::= \text{class } C \text{ extends } C \{ \overline{Cf}; K \overline{M} \}$  *classes*



# Syntax

$P$	$::=$	$\overline{CL} e$	<i>programs</i>
$CL$	$::=$	$\text{class } C \text{ extends } C \{ \overline{Cf}; K \overline{M} \}$	<i>classes</i>
$K$	$::=$	$C(\overline{Cf}) \{ \text{super}(\overline{f}); \overline{\text{this}.f} = \overline{f}; \}$	<i>constructors</i>

# Syntax

$P$	$::=$	$\overline{CL} e$	<i>programs</i>
$CL$	$::=$	$\text{class } C \text{ extends } C \{ \overline{Cf}; K \overline{M} \}$	<i>classes</i>
$K$	$::=$	$C(\overline{Cf}) \{ \text{super}(\overline{f}); \overline{\text{this.f}} = \overline{f}; \}$	<i>constructors</i>
$M$	$::=$	$C m(\overline{Cx}) \{ \text{return } e \}$	<i>methods</i>

# Syntax

$P$	$::=$	$\overline{CL} e$	<i>programs</i>
$CL$	$::=$	$\text{class } C \text{ extends } C \{ \overline{C} f; K \overline{M} \}$	<i>classes</i>
$K$	$::=$	$C(\overline{C} f) \{ \text{super}(\overline{f}); \overline{\text{this}.f} = \overline{f}; \}$	<i>constructors</i>
$M$	$::=$	$C m(\overline{C} x) \{ \text{return } e \}$	<i>methods</i>
$e$	$::=$	$x$	<i>expressions</i>
		$ $ $e.f$	
		$ $ $e.m(\overline{e})$	
		$ $ $\text{new } C(\overline{e})$	
		$ $ $(C) e$	

# Syntax

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

# Syntax

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

# 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()
```



# 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

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

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

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

# Field Lookup

---

$$\overline{fields(Object) = []} \text{ F-OBJECT}$$

# Field Lookup

$$\overline{fields(Object) = []} \quad \text{F-OBJECT}$$

$$\frac{P(C) = \text{class } C \text{ extends } D \{ \overline{C} f; K \overline{M} \} \quad \overline{fields(D) = D g}}{\overline{fields(C) = D g @ \overline{C} f}} \quad \text{F-CLASS}$$

# Method Body Lookup

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

# Method Body Lookup

$$\frac{P(C) = \text{class } C \text{ extends } D \{ \overline{Cf}; K \overline{M} \} \quad B m (\overline{Bx}) \{ \text{return } e \} \in \overline{M}}{mbody(m, C) = (\overline{x}, e)} \quad \text{MB-CLASS}$$

$$\frac{P(C) = \text{class } C \text{ extends } D \{ \overline{Cf}; K \overline{M} \} \quad B m (\overline{Bx}) \{ \text{return } e \} \notin \overline{M}}{mbody(m, C) = mbody(m, D)} \quad \text{MB-SUPER}$$

# Operational Semantics

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



# Operational Semantics

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

$$\frac{e \rightarrow e'}{E[e] \rightarrow E[e']} \text{ E-CONTEXT}$$

# Operational Semantics

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

$$\frac{e \rightarrow e'}{E[e] \rightarrow E[e']} \text{ E-CONTEXT}$$

$$\frac{\text{fields}(C) = \overline{C} f}{\text{new } C(\bar{v}).f_i \rightarrow v_i} \text{ E-PROJ}$$

# Operational Semantics

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

$$\frac{e \rightarrow e'}{E[e] \rightarrow E[e']} \text{ E-CONTEXT}$$

$$\frac{\text{fields}(C) = \overline{C} f}{\text{new } C(\bar{v}).f_i \rightarrow v_i} \text{ E-PROJ}$$

$$\frac{\text{mbody}(m, C) = (\bar{x}, e)}{\text{new } C(\bar{v}).m(\bar{u}) \rightarrow [\bar{x} \mapsto \bar{u}, \text{this} \mapsto \text{new } C(\bar{v})]e} \text{ E-INVK}$$

# Operational Semantics

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

$$\frac{e \rightarrow e'}{E[e] \rightarrow E[e']} \text{ E-CONTEXT}$$

$$\frac{\text{fields}(C) = \bar{C} f}{\text{new } C(\bar{v}).f_i \rightarrow v_i} \text{ E-PROJ}$$

$$\frac{\text{mbody}(m, C) = (\bar{x}, e)}{\text{new } C(\bar{v}).m(\bar{u}) \rightarrow [\bar{x} \mapsto \bar{u}, \text{this} \mapsto \text{new } C(\bar{v})]e} \text{ E-INVK}$$

$$\frac{C \leq D}{(D) \text{new } C(\bar{v}) \rightarrow \text{new } C(\bar{v})} \text{ E-CAST}$$

# Method Type Lookup

$$\frac{\begin{array}{l} P(C) = \text{class } C \text{ extends } D \{ \overline{C} f; K \overline{M} \} \\ B m (\overline{B} x) \{ \text{return } e \} \in \overline{M} \end{array}}{mtype(m, C) = \overline{B} \rightarrow B} \text{ MT-CLASS}$$

# Method Type Lookup

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

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

# Typing Rules

---

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

# Typing Rules

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

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



# Typing Rules

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

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

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

# Typing Rules

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

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

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

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

# Typing Rules

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

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

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

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

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

# Typing Rules

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

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

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

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

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

$$\frac{\Gamma \vdash e : D \quad C \leq D \quad C \neq D}{\Gamma \vdash (C)e : C} \text{ T-DCAST}$$

# Typing Rules

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

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

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

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

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

$$\frac{\Gamma \vdash e : D \quad C \leq D \quad C \neq D}{\Gamma \vdash (C)e : C} \text{ T-DCAST}$$

$$\frac{\Gamma \vdash e : D \quad C \not\leq D \quad D \not\leq C \quad \text{stupid warning}}{\Gamma \vdash (C)e : C} \text{ T-SCAST}$$

# Method Typing

$$\frac{mtype(m, D) = \bar{A} \rightarrow A \text{ implies } \bar{A} = \bar{B} \text{ and } A = B}{\text{override}(m, D, \bar{B} \rightarrow B)} \text{ OVERRIDE}$$

# Method Typing

$$\frac{mtype(m, D) = \bar{A} \rightarrow A \text{ implies } \bar{A} = \bar{B} \text{ and } A = B}{\text{override}(m, D, \bar{B} \rightarrow B)} \text{ OVERRIDE}$$

$$\frac{\begin{array}{l} \overline{x : B}, \text{this} : C \vdash e : A \quad A \leq B \\ P(C) = \text{class } C \text{ extends } D \{ \overline{C} f; K \overline{M} \} \\ \text{override}(m, D, \bar{B} \rightarrow B) \end{array}}{B \ m(\overline{B} \ x) \{ \text{return } e \} \text{ OK in } C} \text{ METHOD-OK}$$

# Class Typing

$$\frac{\begin{array}{l} K = C(\overline{Dg}, \overline{Cf}) \{ \text{super}(\overline{g}); \text{this.f} = \overline{f}; \} \\ \text{fields}(D) = \overline{Dg} \quad \overline{M} \text{ OK in } C \end{array}}{\text{class } C \text{ extends } D \{ \overline{Cf}; K \overline{M} \} \text{ OK}} \text{ CLASS-OK}$$



# 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  $e \rightarrow e'$  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  $e \rightarrow e'$  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  $e \rightarrow e'$ , or*
- 3.  $e = E[(B) \text{ (new } A(\bar{v}))]$  with  $A \not\leq B$ .*