Featherweight Java

Seminar Principles of Programming Languages

Timpe Hörig

Albert-Ludwigs-Universität Freiburg, Germany

Was ist Featherweight Java?

Eine federleichte Version von Java

Warum Featherweight Java?

Formales Modell:

pro: Beweise über Typsicherheit
Überprüfen von Konsequenzen von Erweiterungen/Variationen

aber: Komplexe Sprache = Komplexes Modell

also: Komplexe Sprache zu einer einfachen Sprache vereinfachen
Kompaktheit vs. Vollständigkeit

Featherweight Java

- Klassen mit Vererbung
- Felder
- Methoden
- Ausdrücke:
    1. Objekterstellung
    2. Methodenaufruf
    3. Felderzugriff
    4. Casting
    5. Variable

Überblick

1) Syntax
   - Code Beispiel
   - Abstrakte Syntax
2) Typing
   - Hilfsfunktionen (für das Typing)
   - Subtyping
   - Typsicherheit
3) Reduktions-Regeln
   - Reihenfolge der Auswertung von Ausdrücken
   - Berechnung
   - Kongruenz

Syntax

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

((Pair)new Pair(new Pair(new A(), new B()), new A()).fst).snd
```

Abstrakte Syntax

L ::= class C extends C‘ {$\bar{C}\ \bar{f}$; K $\bar{M}$ }

K ::= C($\bar{C}\ \bar{f}$){ super($\bar{f}$); this.$\bar{f}$=$\bar{f}$; }

M ::= C m($\bar{C}\ \bar{x}$){ return e; }

e ::= x | e.f | e.m(ē) | new C(ē) | (C)ē

Abstrakte Syntax | Klassen

L ::= class C extends C‘ {$\bar{C}\ \bar{f}$; K $\bar{M}$ }

```
class Pair extends Object {
    Object fst;
    Object snd;
    $K$
    $M_1$
     …
    $M_n$
}
```

Abstrakte Syntax | Konstruktor

$$\mathrm{K} \ ::= \ \mathrm{C}(\bar{C}\ \bar{f})\{\ \mathrm{super}(\bar{f});\ \mathrm{this}.\bar{f}=\bar{f};\ \}$$

```
Pair(Object fst, Object snd) {
        super();
        this.fst=fst;
        this.snd=snd;
}
```

Abstrakte Syntax | Methode

$$M \ ::= \ C\ m(\bar{C}\ \bar{x})\{\ \text{return } e;\ \}$$

```
Pair setfst(Object newfst) {
        return new Pair(newfst, this.snd);
}
```

Abstrakte Syntax | Ausdruck

$$e \;::=\; x \mid e.f \mid e.m(\bar{e}) \mid \text{new } C(\bar{e}) \mid (C)e$$

1. this | newfst
2. this.fst
3. this.setfst(new A())
4. new Pair(new A(), new B())
5. (Pair)new Pair(new A(), new A())

Hilfsfunktionen | Feldersuche

$$fields(Object) = \bullet$$

$$\frac{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ K\ \bar{M}\} \qquad fields(D) = \bar{D}\ \bar{g}}{fields(C) = \bar{D}\ \bar{g}, \bar{C}\ \bar{f}}$$

Hilfsfunktionen | Feldersuche - Beispiel

$$fields(Object) = \bullet$$

$$\frac{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ K\ \bar{M}\} \qquad fields(D) = \bar{D}\ \bar{g}}{fields(C) = \bar{D}\ \bar{g}, \bar{C}\ \bar{f}}$$

```
class D extends Object {
   Object field_of_D;
   D(Object field_of_D) {
      super();
      this.field_of_D=field_of_D;
   }
}

class C extends D {
   Object field_of_C;
   C(Object field_of_C, Object field_of_D) {
      super(field_of_D);
      this.field_of_C=field_of_C;
   }
}
```

$fields(C) = fields(C)$

$fields(C) =$ Object field_of_C, $fields(D)$

$fields(C) =$ Object field_of_C, Object field_of_D, $fields(Object)$

$fields(C) =$ Object field_of_C, Object_field_of_D

Hilfsfunktionen | Methodentypsuche

$$\frac{class\ C\ extends\ D\ \left\{\bar{C}\ \bar{f};\ K\ \bar{M}\right\} \qquad B\ m(\bar{B}\ \bar{x})\ \{\ return\ e;\ \} \in \bar{M}}{mtype(m, C) = \bar{B} \rightarrow B}$$

$$\frac{class\ C\ extends\ D\ \left\{\bar{C}\ \bar{f};\ K\ \bar{M}\right\} \qquad m \notin \bar{M}}{mtype(m, C) = mtype(m, D)}$$

Typing | Subtyping

$$C <: C$$

$$\frac{C\ <:\ D \quad D\ <:\ E}{C\ <:\ E}$$

$$\frac{class\ C\ extends\ D\ \{\ldots\}}{C\ <: D}$$

Typing | Ausdruck | Variable

$$\Gamma \vdash \mathrm{x} : \Gamma(\mathrm{x})$$

Typing | Ausdruck | Variable - Beispiel

$$\Gamma \vdash \mathrm{x} : \Gamma(\mathrm{x})$$

$\Gamma_1 =$

```
class A extends Object {
    Object field;
    A(Object field) { super(); this.field=field; }
    A set_field(Object new_field) {
        return new A(new_field);
    }
}
```

$\Gamma_1 \vdash \text{new\_field} : \text{Object}$

$\Gamma_2 =$

```
class B extends Object {
    A field;
    B(Object field) { super(); this.field=field; }
    B set_field(A new_field) {
        return new B(new_field);
    }
}
```

$\Gamma_2 \vdash \text{new\_field} : \text{A}$

Typing | Ausdruck | Feldzugriff, Methodenaufruf, Objekterstellung

$$\frac{\Gamma \vdash e_0 : C_0 \qquad fields(C_0) = \bar{C}\ \bar{f}}{\Gamma \vdash e_0.f_i : C_i}$$

$$\frac{\Gamma \vdash e_0 : C_0 \qquad mytype(m, C_0) = \bar{D} \to C \qquad \Gamma \vdash \bar{\mathrm{e}} : \bar{C} \qquad \bar{C} <: \bar{D}}{\Gamma \vdash e_0.m(\bar{\mathrm{e}}) : C}$$

$$\frac{fields(C) = \bar{D}\ \bar{f} \qquad \Gamma \vdash \bar{\mathrm{e}} : \bar{C} \qquad \bar{C} <: \bar{D}}{\Gamma \vdash new\ C(\bar{\mathrm{e}}) : C}$$

Typing | Ausdruck | Casting (Up – Down – Stupid)

$$\frac{\Gamma \vdash e_0 : D \qquad D <: C}{\Gamma \vdash (C)e_0 : C}$$

$$\frac{\Gamma \vdash e_0 : D \qquad C <: D \qquad C \neq D}{\Gamma \vdash (C)e_0 : C}$$

$$\frac{\Gamma \vdash e_0 : D \qquad C \not<: D \qquad D \not<: C \qquad stupid\ warning}{\Gamma \vdash (C)e_0 : C}$$

Typing | Methode

$$\frac{\bar{x} : \bar{C},\ this :\ \mathrm{C} \vdash e_0 : E_0 \qquad E_0 <: C_0 \\ class\ C\ extends\ D\ \{\ldots\} \\ if\ mtype(m, D) = \overline{D} \rightarrow D_0,\ then\ \bar{C} = \overline{D}\ and\ C_0 = D_0}{C_0\ m(\bar{C}\ \bar{x})\{\ return\ e_0;\}\ OK\ IN\ C}$$

Typing | Methode - Beispiel

$$\frac{\begin{array}{c}\bar{x} : \bar{C},\ this :\ \mathrm{C} \vdash e_0 : E_0 \qquad E_0 <: C_0 \\ class\ C\ extends\ D\ \{\ldots\} \\ if\ mtype(m, D) = \overline{D} \rightarrow D_0,\ then\ \bar{C} = \overline{D}\ and\ C_0 = D_0\end{array}}{C_0\ m(\bar{C}\ \bar{x})\{\,return\ e_0;\}\ OK\ IN\ C}$$

```
class C extends D {
    C() { super(); }
    C_0 m(C_1 x_1, C_2 x_2) {
        return e_0;
    }
}

class D extends Object {
    D() { super(); }
    D_0 m(D_1 x_1, D_2 x_2) {
        return e_0;
    }
}
```

Reduktions-Regeln | Berechnung (Feldzugriff, Methodenaufruf, Casting)

$$\frac{fields(C) = \bar{C}\ \bar{f}}{\left(new\ C(\bar{\mathrm{e}})\right).f_i\ \rightarrow e_i}$$

$$\frac{mbody(m, C) = \bar{x}.e_0}{\left(new\ C(\bar{\mathrm{e}})\right).m\left(\bar{d}\right) \rightarrow \left[\frac{\bar{d}}{\bar{x}}, \frac{new\ C(\bar{e})}{this}\right] e_0}$$

$$\frac{C <: D}{(D)\left(new\ C(\bar{\mathrm{e}})\right) \rightarrow new\ C(\bar{\mathrm{e}})}$$

Reduktions-Regeln| Berechnung - Feldzugriff

$$\boxed{\dfrac{fields(C) = \bar{C}\ \bar{f}}{\left(new\ C(\bar{\mathrm{e}})\right).f_i\ \rightarrow e_i}}$$

$$fields(C) = C_0\ f_0, \dots, C_\mathrm{n}\ f_\mathrm{n}$$

$$\left(new\ C(e_0, \dots, e_i, \dots e_n)\right).f_i\ \rightarrow e_i$$

```
class C extends Object {
    $C_0\ f_0$;
     …
    $C_\mathrm{n}\ f_\mathrm{n}$;
    C($C_0\ f_0, \dots, C_\mathrm{n}\ f_\mathrm{n}$) {
        super();
        this.$f_0$=$f_0$;
         …
        this.$f_\mathrm{n}$=$f_\mathrm{n}$;
    }
}
```

Reduktions-Regeln | Kongruenz

$$\frac{e_0 \to {e_0}'}{e_0.f \;\to e_0'.f}$$

$$\frac{e_i \to {e_i}'}{new\ C(\ldots, e_i, \ldots) \to new\ C(\ldots, e_i', \ldots)}$$

$$\frac{e_0 \to {e_0}'}{e_0.m(\bar{e}) \to e_0'.m(\bar{e})}$$

$$\frac{e_i \to {e_i}'}{(C)e_0 \to (C){e_0}'}$$

$$\frac{e_i \to {e_i}'}{e_0.m(\ldots, e_i, \ldots) \to e_0.m(\ldots, e_i', \ldots)}$$

Featherweight Java

Seminar Principles of Programming Languages

Vielen Dank für Ihre Aufmerksamkeit

Timpe Hörig

Albert-Ludwigs-Universität Freiburg, Germany

Quelle

Featherweight Java: a minimal core calculus for Java and GJ

Atsushi Igarashi, Benjamin C. Pierce, and Philip Wadler. 2001. Featherweight Java: a minimal core calculus for Java and GJ. ACM Trans. Program. Lang. Syst. 23, 3 (May 2001), 396–450. https://doi.org/10.1145/503502.503505

Hilfsfunktionen |Method body lookup

$$\frac{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ K\ \bar{M}\} \qquad B\ m(\bar{B}\ \bar{x})\ \{\ return\ e;\ \} \in \bar{M}}{mbody(m, C) = \bar{x}.e}$$

$$\frac{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ K\ \bar{M}\} \qquad m \notin \bar{M}}{mbody(m, C) = mbody(m, D)}$$

Typing |Class

$$\frac{K = C\left(\bar{D}\ \bar{g},\ \bar{C}\ \bar{f}\right)\left\{ super(\bar{g});\ this.\bar{f} = \bar{f}; \right\} \qquad fields(D) = \bar{D}\ \bar{g} \qquad \bar{M}\ \text{OK IN}\ C}{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ \text{K}\ \bar{M}\ \}\ OK}$$