Questions?

Top
Java-Like Language: Semantics

• Values: ints, objects

• How should we represent an object?

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>5</td>
</tr>
<tr>
<td>getx()</td>
<td>return x;</td>
</tr>
<tr>
<td>sety(n)</td>
<td>y := n;</td>
</tr>
</tbody>
</table>

Point(3, 5)
Java-Like Language: Semantics

• Values: ints, objects

• How should we represent an object?

  Point(3, 5)  Square(4)  Item("book", 15.55, 200)

• In general: $C(v_1, v_2, \ldots, v_n)$ where $C$ is the object’s class, $v_1$ is the value of its first field, $v_2$ is the value of its second field, etc.
  — starting with fields inherited from superclasses!
Java-Like Language: Semantics

\[
\frac{(e_1, \sigma) \downarrow v_1 \ldots (e_n, \sigma) \downarrow v_n}{(\text{new } C(e_1, \ldots, e_n), \sigma) \downarrow C(v_1, \ldots, v_n)}
\]

\[
\frac{(e, \sigma) \downarrow C(v_1, \ldots, v_n) \quad \text{fields}(C) = \tau_1 f_1, \ldots, \tau_n f_n}{(e.f_i, \sigma) \downarrow v_i}
\]
Java-Like Language: Semantics

\[
\begin{align*}
(e_1, \sigma) \downarrow v_1 \quad \ldots \quad (e_n, \sigma) \downarrow v_n \\
\text{(new } C(e_1, \ldots, e_n), \sigma) \downarrow C(v_1, \ldots, v_n)
\end{align*}
\]

\[
(e, \sigma) \downarrow C(\bar{v}) \quad \text{fields}(C)[i] = \tau f \\
(e.f, \sigma) \downarrow v[i]
\]
Functions: Semantics of Calls

\[
\begin{align*}
(e_1, \sigma) \Downarrow v_1 \quad & \ldots \quad (e_n, \sigma) \Downarrow v_n \quad \sigma(f) = f(x_1, \ldots, x_n) \{ c \} \\
(x := f(e_1, \ldots, e_n), k, \sigma) \rightarrow \\
(c, (\sigma, x) :: k, \sigma[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n])
\end{align*}
\]

- Evaluate the arguments \( e_1, \ldots, e_n \)
- Look up \( f \) in \( \sigma \)
- Execute the body of \( f \) and produce a return value
- Assign the return value to \( x \)
OO: Semantics of Methods

\[(e_1, \sigma) \downarrow v_1 \quad \ldots \quad (e_n, \sigma) \downarrow v_n \quad \sigma(f) = f(x_1, \ldots, x_n)\{c\}\]

\[(x := e.m(e_1, \ldots, e_n), k, \sigma) \rightarrow (c, (\sigma, x) :: k, \sigma[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n])\]

• Evaluate the arguments \(e_1, \ldots, e_n\)
• Look up \(f\) in \(\sigma\)
• Execute the body of \(f\) and produce a return value
• Assign the return value to \(x\)
(e_1, \sigma) \downarrow v_1 \quad \ldots \quad (e_n, \sigma) \downarrow v_n \quad \sigma(f) = f(x_1, \ldots, x_n)\{c\}

\begin{align*}
(x := e.m(e_1, \ldots, e_n), k, \sigma) & \rightarrow \\
(c, (\sigma, x) \:: k, \sigma[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n])
\end{align*}

• Evaluate the arguments \(e_1, \ldots, e_n\) and the object \(e\)
• Look up \(m\) in the methods of \(e\)'s class
• Execute the body of \(m\) (with \textit{this} set to \(e\)) and produce a return value
• Assign the return value to \(x\)
• Evaluate the arguments $e_1, ..., e_n$ and the object $e$
• Look up $m$ in the methods of $e$’s class
• Execute the body of $m$ (with this set to $e$) and produce a return value
• Assign the return value to $x$
### OO: Semantics of Methods

\[
(e, \sigma) \downarrow C(u_1, \ldots, u_n) \quad (e_1, \sigma) \downarrow v_1 \quad \ldots \quad (e_n, \sigma) \downarrow v_n
\]

\[
\text{methods}(C) = \cdots, \tau m(\tau_1 \ x_1, \ldots, \tau_n \ x_n)\{ \ c \ \}
\]

\[
(x := e. m(e_1, \ldots, e_n), k, \sigma) \rightarrow
\]

\[
(c, (\sigma, x) :: k, \sigma[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n])
\]

- Evaluate the arguments \(e_1, \ldots, e_n\) and the object \(e\)
- Look up \(m\) in the methods of \(e\)'s class
- Execute the body of \(m\) (with \textbf{this} set to \(e\)) and produce a return value
- Assign the return value to \(x\)
(e, \sigma) \downarrow C(u_1, ..., u_n) \quad (e_1, \sigma) \downarrow v_1 \quad ... \quad (e_n, \sigma) \downarrow v_n

\text{methods}(C) = \ldots, \tau \ m(\tau_1 \ x_1, ..., \tau_n \ x_n)\{ \ c \ \}

\frac{(x := e. \ m(e_1, ..., e_n), k, \sigma)}{(c, (\sigma, x) :: k, \sigma[\text{this} = C(u_1, ..., u_n), x_1 \mapsto v_1, ..., x_n \mapsto v_n])}

- Evaluate the arguments \( e_1, ..., e_n \) and the object \( e \)
- Look up \( m \) in the methods of \( e \)'s class
- Execute the body of \( m \) (with \( \text{this} \) set to \( e \)) and produce a return value
- Assign the return value to \( x \)
Java-Like Language: Contexts

• In our simple imperative language, declarations could be stored in the type context and the runtime state

\[
\Gamma[x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n] \vdash c : \text{ok}
\]

\[
\Gamma \vdash \tau f(\tau_1 x_1, \ldots, \tau_n x_n)\{c\} : \Gamma[f \mapsto \tau(\tau_1 x_1, \ldots, \tau_n x_n)]
\]

\[
(\tau f(\tau_1 x_1, \ldots, \tau_n x_n)\{c\}, \sigma) \rightarrow (\text{skip}, \sigma[f \mapsto f(x_1, \ldots, x_n)\{c\}])
\]

• In Java, both typing and semantics need the full declaration of each class

• We implicitly store class declarations in a class table
Java-Like Language: Contexts

- In Java, both typing and semantics need the full declaration of each class
- We implicitly store class declarations in a class table
- Java uses type information at runtime!
  - Every object is tagged with its class
  - The class determines which fields and methods are defined
  - The class determines which version of a method is called
- Whereas in C, types disappear at runtime
Java-Like Language: Contexts

• In Java, both typing and semantics need the full declaration of each class

• We implicitly store class declarations in a class table

\[
(e, \sigma) \downarrow \text{new } C(v_1, \ldots, v_n) \quad \text{fields}(C)[i] = \tau f \\
(e.f, \sigma) \downarrow v_i
\]
Java-Like Language: Contexts

• In Java, both typing and semantics need the full declaration of each class
• We implicitly store class declarations in a class table

\[
(e, \sigma) \downarrow \text{new } C(v_1, \ldots, v_n) \quad \text{fields}(CT, C)[i] = \tau f \\
(e.f, \sigma) \downarrow v_i
\]

• \text{CT} is a global structure used by both the type system and the semantics, and computed in preprocessing
Questions?

Top
Java-Like Language: Casts

Building build(Building model);

class School extends Building

School s = new School();
s2 = w.build((Building) s);
((School) s2).getCourse();

• Exercise: When can we cast from one class to another?
Java-Like Language: Casts

Building build(Building model);

class School extends Building

School s = new School();
s2 = w.build((Building) s);  // upcast from School to Building
((School) s2).getCourse();  // downcast from Building to School
Java-Like Language: Casts

- Upcast: always safe, doesn’t do anything
- Downcast: safe only if object actually has the right type – we might not know until runtime
- Other casts: ??

```
B b = new B();
A a = (A) b;
\[?\]
\[\Gamma \vdash (C) e : C\]
```
Java-Like Language: Casts

• Upcast: always safe, doesn’t do anything
• Downcast: safe only if object actually has the right type – we might not know until runtime
• Other casts: ??

B b = new B();
A a = (A) ((Object) b);
Java-Like Language: Casts

• Upcast: always safe, doesn’t do anything
• Downcast: safe only if object actually has the right type – we might not know until runtime
• Other casts: ??

B b = new B();
A a = (A) ((Object) b);

\[ \begin{align*}
\Gamma \vdash e : D & \quad D <: C \text{ or } C <: D \\
\Gamma \vdash (C) e : C
\end{align*} \]
Java-Like Language: Casts

• Upcast: always safe, doesn’t do anything
• Downcast: safe only if object actually has the right type
• Other casts: ??

```java
B b = new B();
A a = (A) ((Object) b);
```

$$
\begin{align*}
(e, \sigma) \downarrow C(v_1, \ldots, v_n) & \quad C <: D \\
((D) e, \sigma) \downarrow C(v_1, \ldots, v_n)
\end{align*}
$$

$$
\begin{align*}
(e, \sigma) \downarrow C(v_1, \ldots, v_n) & \quad \text{not } C <: D \\
((D) e, \sigma) \downarrow \text{ClassCastException}
\end{align*}
$$
Java-Like Language: Syntax

$$CL ::= \text{class }<\text{id}> \text{ extends }<\text{id}> \{ \ T \ <\text{id}> , \,..., \ T \ <\text{id}> ; \ K \ M \ ... \ M \ \}$$

$$K ::= \ <\text{id}> ( \ T \ <\text{id}> , \,..., \ T \ <\text{id}> ) \{}$$

$$\text{super(} <\text{id}> , \,..., \ <\text{id}> \text{);} \ this. <\text{id}> = <\text{id}> ; \,...; \ this. <\text{id}> = <\text{id}> ; \}$$

$$M ::= \ T \ <\text{id}> ( \ T \ <\text{id}> , \,..., \ T \ <\text{id}> ) \{ \ C \}$$

$$E ::= \ ... \ | \ E. <\text{id}>$$

$$C ::= \ ... \ | \ <\text{id}> = E. <\text{id}> ( E, \,..., \ E ) ; \ | \ <\text{id}> = \text{new} \ <\text{id}> ( E, \,..., \ E )$$

$$T ::= \text{int} \ | \ <\text{id}>$$

$$P ::= CL \ ... \ CL \ C$$
**Java-Like Language: Syntax**

\[ CL ::= \text{class <id> extends <id> \{ T <id>, ..., T <id>; K M ... M \} } \]

\[ K ::= \text{id}(T <id>, ..., T <id>)\{ \]

\[ \quad \text{super(<id>, ..., <id>); this.<id> = <id>; ...; this.<id> = <id>; } \]

\[ M ::= T <id>(T <id>, ..., T <id>)\{ C \} \]

\[ E ::= ... \mid E.<id> \]

\[ C ::= ... \mid <id> = E.<id>(...) \mid <id> = \text{new <id>(...)} \mid <id> = E \mid E.<id> = E \]

\[ T ::= \text{int} \mid <id> \]

\[ P ::= CL \ldots CL C \]
Objects vs. Values

• We said “objects are values”

Objects:
new Point(3, 5)
can be stored in variables
have pieces that can change
different objects can have the same values in them

Values:
5, true, etc.
can be stored in variables
can’t change
if the value is the same, they’re equal
Objects vs. Values

• We said “objects are values”, but they’re also like variables!

**Objects:**
- new Point(3, 5)
- can be stored in variables
- have pieces that can change
- different objects can have the same values in them

**Values:**
- 5, true, etc.
- can be stored in variables
- can’t change
- if the value is the same, they’re equal
Java-Like Language: Mutable Objects

• Our language so far has no field-set operation!

A a1 = new A(3, 5);
A a2 = a1;
a1.x = 4;
int result = a2.x; // should be 4
Java-Like Language: Mutable Objects

• Our language so far has no field-set operation!

A a1 = new A(3, 5);  // {a1 = A(3, 5)}
A a2 = a1;
a1.x = 4;
int result = a2.x;  // should be 4
Java-Like Language: Mutable Objects

• Our language so far has no field-set operation!

A a1 = new A(3, 5);              {a1 = A(3, 5)}
A a2 = a1;                      {a1 = A(3, 5), a2 = A(3,5)}
a1.x = 4;
int result = a2.x; // should be 4
Java-Like Language: Mutable Objects

• Our language so far has no field-set operation!

```
A a1 = new A(3, 5);          {a1 = A(3, 5)}
A a2 = a1;                    {a1 = A(3, 5), a2 = A(3,5)}
a1.x = 4;                     {a1 = A(4, 5), a2 = A(3,5)}
int result = a2.x; // should be 4
```

• Exercise: How could we write semantics where a2.x returns 4?
Java-Like Language: Mutable Objects

• Our language so far has no field-set operation!

A a1 = new A(3, 5);  {a1 = r1, r1 -> A(3, 5)}
A a2 = a1;            {a1 = r1, a2 = r1, r1 -> A(3, 5)}
a1.x = 4;             {a1 = r1, a2 = r1, r1 -> A(4, 5)}
int result = a2.x;   // should be 4
a1 = new A(6, 7);     {a1 = r2, a2 = r1, r1 -> A(4, 5), r2 -> A(6, 7)}

• Two-level model: variables hold references, references point to values
Java-Like Language: Mutable Objects

• Split the state $\sigma$ into two levels
• Program state is now a tuple $(c, k, \rho, \sigma)$ where:
  — $c$ is the currently executing command
  — $k$ is the call stack
  — $\rho$ is the environment, mapping variables to either primitive values (int, bool) or references
  — $\sigma$ is the store, mapping references to object values
Java-Like Language: Semantics

\[(e, \sigma) \Downarrow C(v_1, \ldots, v_n) \quad \text{fields}(C) = \tau_1 f_1, \ldots, \tau_n f_n \]

\[ (e.f_i, \sigma) \Downarrow v_i \]
Java-Like Language: Semantics

\[
\begin{align*}
(e, \rho, \sigma) & \downarrow r \quad \sigma(r) = C(\vec{v}) \quad \text{fields}(C)[i] = \tau f \\
& \quad \frac{e. f, \rho, \sigma \downarrow v[i]}{v[i]}
\end{align*}
\]

\[
\begin{align*}
\rho(x) &= v \\
& \quad \frac{(x, \rho, \sigma) \downarrow v}{v}
\end{align*}
\]

\[
\begin{align*}
(e, \rho, \sigma) & \downarrow v \\
& \quad \frac{(x := e, \rho, \sigma) \rightarrow (\text{skip}, \rho[x \mapsto v], \sigma)}{(x := e, \rho, \sigma) \rightarrow (\text{skip}, \rho[x \mapsto v], \sigma)}
\end{align*}
\]
Java-Like Language: Semantics

\[
\frac{(e_1, \sigma) \Downarrow v_1 \ldots (e_n, \sigma) \Downarrow v_n}{(\text{new } C(e_1, \ldots, e_n), \sigma) \Downarrow C(v_1, \ldots, v_n)}
\]
Java-Like Language: Semantics

\[
\begin{align*}
(e_1, \rho, \sigma) \downarrow v_1 & \quad \ldots \quad (e_n, \rho, \sigma) \downarrow v_n \quad r \not\in \text{dom}(\sigma) \\
& \quad \frac{\text{(x = new } C(e_1, \ldots, e_n), \rho, \sigma)}{(\text{skip, } \rho[\text{x }\mapsto r], \sigma[r \mapsto C(v_1, \ldots, v_n)])}
\end{align*}
\]

\[
\begin{align*}
(e, \rho, \sigma) \downarrow r & \quad \sigma(r) = C(\vec{v}) \\
\text{fields}(C)[i] = \tau f & \quad (e_1, \rho, \sigma) \downarrow v \\
& \quad \frac{(e. f := e_1, \rho, \sigma)}{(\text{skip, } \rho, \sigma[r \mapsto C(\vec{v}[i \mapsto v])])}
\end{align*}
\]
Java-Like Language: Mutable Objects

A a1 = new A(3, 5);  \{a1 = r1, r1 -> A(3, 5)}
A a2 = a1;  \{a1 = r1, a2 = r1, r1 -> A(3, 5)}
a1.x = 4;  \{a1 = r1, a2 = r1, r1 -> A(4, 5)}
int result = a2.x; // should be 4
a1 = new A(6, 7);  \{a1 = r2, a2 = r1, r1 -> A(4, 5), r2 -> A(6, 7)}
Questions?