CS 444 - Compiler Construction **Winter 2020**

Lecture 11: February 10th , 2020

Lecturer: Ondřej Lhoták *Notes By: Harsh Mistry*

11.1 Context-sensitive analysis continued

11.1.1 Name Resolution

- Link each use of a name to a declaration
- In Java : A2 and A3
  1. Build global environment
  2. Resolve type names
  3. Build/check class hierarchy (methods/files)
  4. Disambiguate ambiguous namespace
     - In Java you can't simply determine the namespace based on location of usage
  5. Resolve "expressions" (Variables, static fields)
  6. Type checking
  7. Resolve methods instance (non-static) fields
- (1) Building a global environment
  - Simply create a set of all classes with packages
  - Its important to note that the "default" package is NOT the root package
  - There is no way to declare class that falls within the default package
- (2) Resolving type names (Refer to JLS 6.5)
  - Types
    - Qualified Name (with dots) (i.e a.b.c)
      - Must be full name of type
      - No notion of relative path names
    - Simple Name (no dot) (c)
  - Resolving steps
    1. Is it the enclosing class/interface?
    2. Is it a single-type import (a.b.c)?
    3. Is it a type in the current package (of the enclosing type)?
    4. Is it a import-on-demand package (a.b.t)?
  - Each step ① - ④ must be unambiguous

- (3) Building and checking class hierarchy
  - Simple Checks
    1. Class A extends B $\implies$ B must be class (Refer to JLS 8.1.3)
    2. Class A implements D *implies* D must be an interface (Refer to JLS 8.1.4)
    3. No duplicate interfaces (i.e `Class E implements F,F`) (Refer to JLS 8.1.4)
    4. B cannot be final (Refer to JLS 8.1.3)
    5. Two constructors of the same class must have different signatures (Parameter Lists) (Refer to JLS 8.8.2)

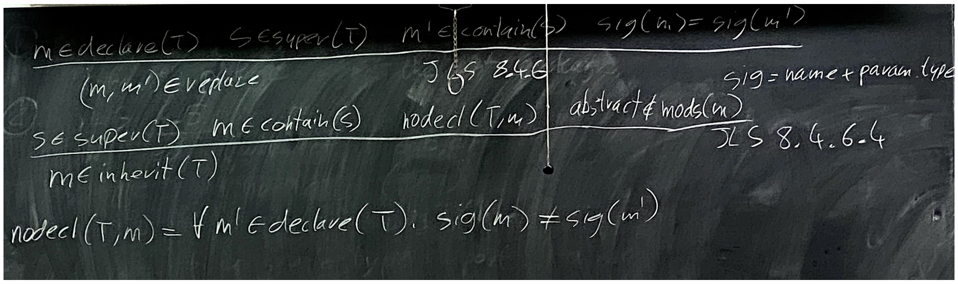
    Definition 11.1 *Super(A) = direct super-classes / interfaces of A*
    Examples

    - *Class A extends B implements C,D,E* $\implies$ *super(A)* $= \{B, C, D, E\}$
    - *If unspecified, B is Java.lang.Object*
    - *super(Java.lang.Object)* $= \{\}$
    - *Interface F extends GHI*
    - *super(F)* $= \{G, H, I\}$
  - Rules
    - $$\frac{T \in super(S)}{S < T}$$
    - $$\frac{S < T}{S \leq T}$$
    - $$\frac{}{T \leq T}$$
    - $$\frac{S < T' \quad T' < T}{S < T}$$
    - $$S < T \implies \text{S is a } \underline{\text{strict subtype}} \text{ of T}$$

    Definition 11.2 -

    - *declare(T) = set of methods/fields in body of T*
    - *inherit(T) = m/f that T inherits*
    - *contain(T) = declare(T)* $\cup$ *inherit(T)*
    - *replace*(m, m')
      - m *"overrides"* m'
      - m *"hides"* m'
      - f *"hides"* f'

- More Rules



$$\frac{m \in declare(T) \quad S \in super(T) \quad m' \in contain(S) \quad sig(m) = sig(m')}{(m, m') \in replace}$$

JLS 8.4.8

sig = name + param type

$$\frac{S \in super(T) \quad m \in contain(S) \quad nodecl(T, m) \quad abstract \notin mods(m)}{m \in inherit(T)}$$

JLS 8.4.6.4

$$nodecl(T, m) = \forall m' \in declare(T).\ sig(m) \neq sig(m')$$