SWEN430 - Compiler Engineering

Lecture 7 - Typing I

Lindsay Groves & David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

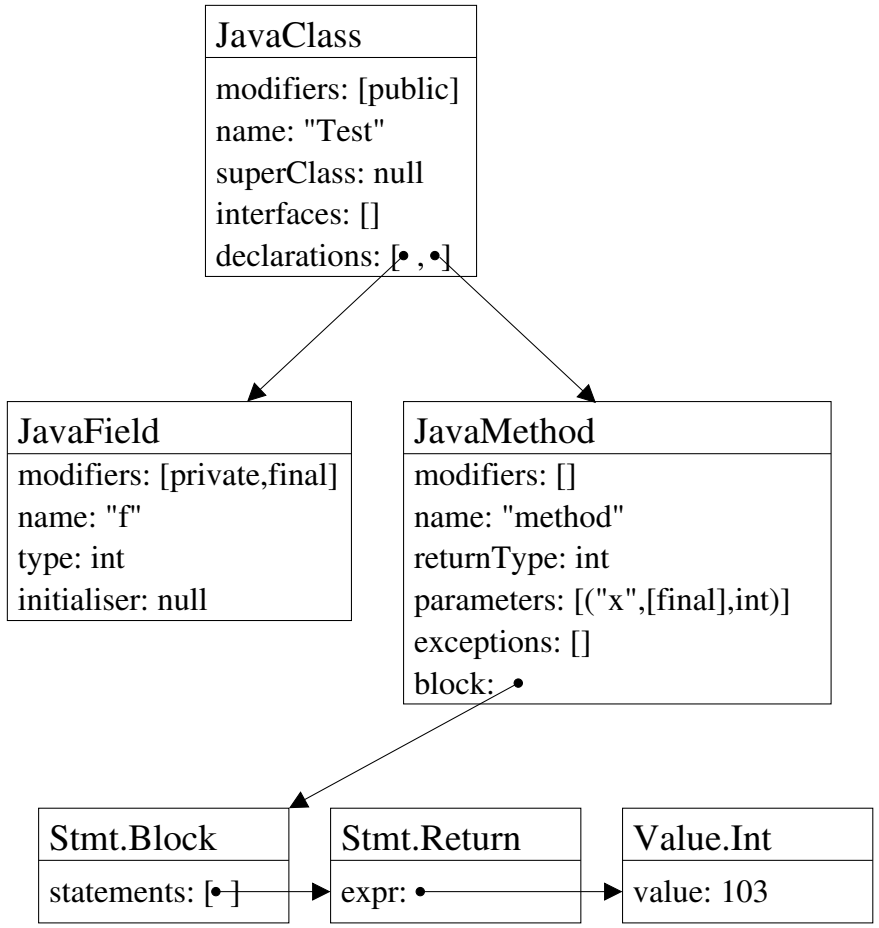
Building an Abstract Syntax Tree (AST)

- Easy to extend a recursive descent parser to build an AST.
- Each method called builds and returns an AST for the thing it parses.
- The method builds its AST from the ASTs for its components.
- So the whole tree is built bottom up.

Abstract Syntax Tree (AST)

```
public class Test {
 private final int f;

 int method(int final x) {
  return 103;
}}
```



Error Checking

- The parser only checks context free syntax.
- Lots of inputs it accepts are not valid programs.
- Need extra checks to eliminate them:
  - Context conditions: Constraints on where symbols can be used.
  - Type checking: Ensuring operations are applied to arguments of the correct type.
  - Static analysis: Checking (some cases of) semantic conditions such as definite assignment and unreachable code.

Context Checking

- Variables, types and methods must be declared
- ... and can only be declared once in a given scope
- Methods must be called with the right number of arguments
- Break and continue can only appear within a loop (or switch)
- Switch case labels must be distinct

Ex: What context conditions (should) apply to While programs?

Ex: Where should they be checked??

Unresolved Variables

```
class Test {
 int x;

 class Inner { int f() { return x; } }

 int f(int r) {
  { int y; r = r + y; }
  { int z; }
  r = r + z;
  return r;
}}
```

Example:

- Access to `x` should resolve to a non-local field access
- Access to `y` should resolve to a local variable
- Access to `z` should cause syntax error

Unresolved Variables

- Are placeholders for variable accesses
- Are converted into field, local or non-local accesses later

Type Checking in While

- Condition in while and if must be Boolean.
- Operands of +, - , * and / must be numeric.
- Operands on && and || must be Boolean.
- In $e_1[e_2]$, e_1 must be an array and $e + 2$ an interger.
- In $e.f$, e must be a record with field f.

Ex: What others?

Easy to check these: Just traverse the AST performing required checks.

Java Type Checking

The Java compiler must check:

- That primitive types are used correctly
- That reference types are used correctly
- That methods and fields exist with appropriate types
- That method overriding respects modifiers
- That generic types are used correctly
- That wildcard types are used correctly
- ...

```
class Test implements Inter <? extends Comparable> {
 double f(float f) {
  int x = (int) f;
  return f;
 }
 void g(Test x, Inter <? extends Comparable> y) {
  y = x; // up cast
  x = (Test) y; // down cast
 }
}
```

Java Binary Numeric Conversion

JLS 5.6.2 Binary Numeric Promotion:

When an operator applies binary numeric promotion to a pair of operands, each of which must denote a value that is convertible to a numeric type, the following rules apply, in order, using widening conversion (Section 5.1.2) to convert operands as necessary:

- *If any of the operands is of a reference type, unboxing conversion (Section 5.1.8) is performed. Then:*
- *If either operand is of type double, the other is converted to double.*
- *Otherwise, if either operand is of type float, the other is converted to float.*
- *Otherwise, if either operand is of type long, the other is converted to long.*
- *Otherwise, both operands are converted to type int.*

```
int i = 2;
float f = 1.0f;
double d = (i * f) * 2.0;
```