Compiler Construction
Winter 2020

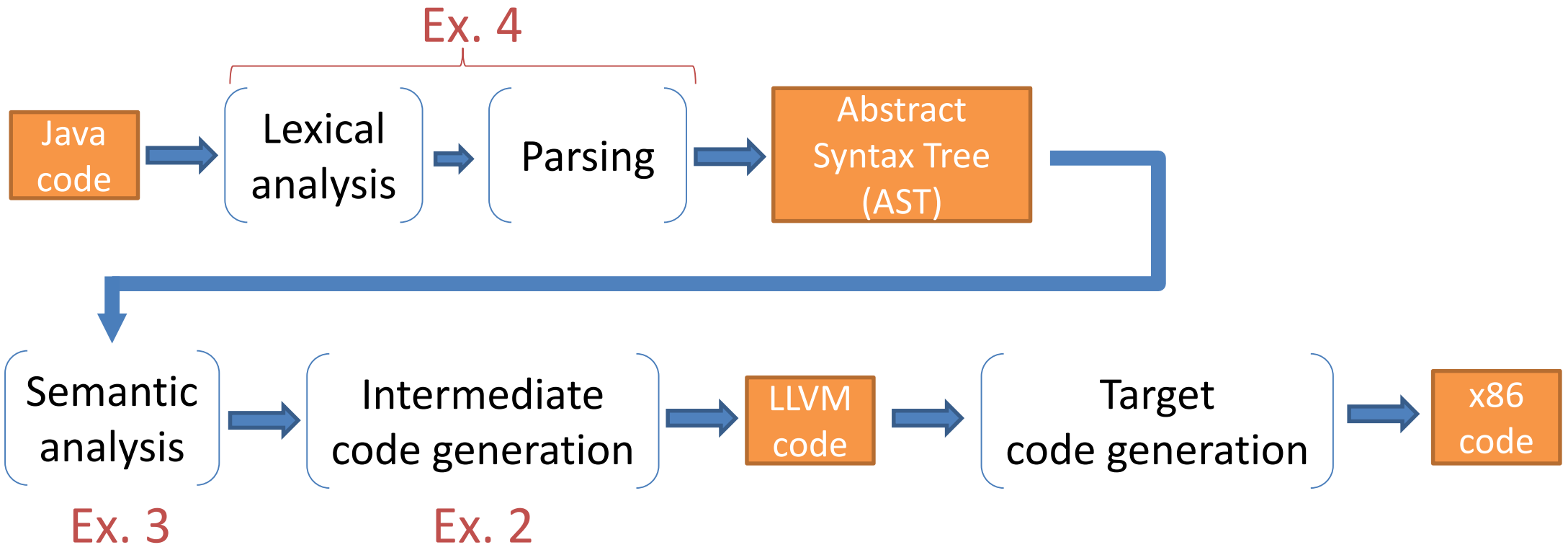
Recitation 1:
Abstract Syntax of MiniJava

Yotam Feldman

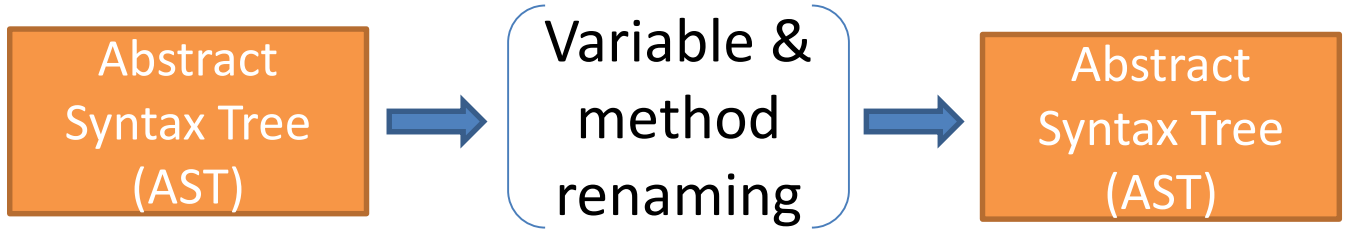
Administrative

- Course website
  - Lectures: https://www.cs.tau.ac.il/~msagiv/courses/wcc20.html
  - Recitations and project: http://www.cs.tau.ac.il/research/yotam.feldman/courses/wcc20/wcc20.html
  - Forums in Moodle
- Grade: 50% project, 50% exam
  - 4 programming assignments
  - Assignments in groups of **3**
- My reception hour: Sunday, 14-15
  - Email in advance, yotamfe1@mail.tau.ac.il
  - (**Not** @tauex)
- COVID19
  - I very much (!) appreciate you opening your webcams, but this is **not** required

Compilation Phases



Ex. 1:



MiniJava

- A subset of Java
- Many restrictions
  - See project website

Simple Example.java

```
class Main {
    public static void main(String[] a) {
        System.out.println(new Simple().Start(1, 2));
    }
}

class Trivial {
        int f;
}
```

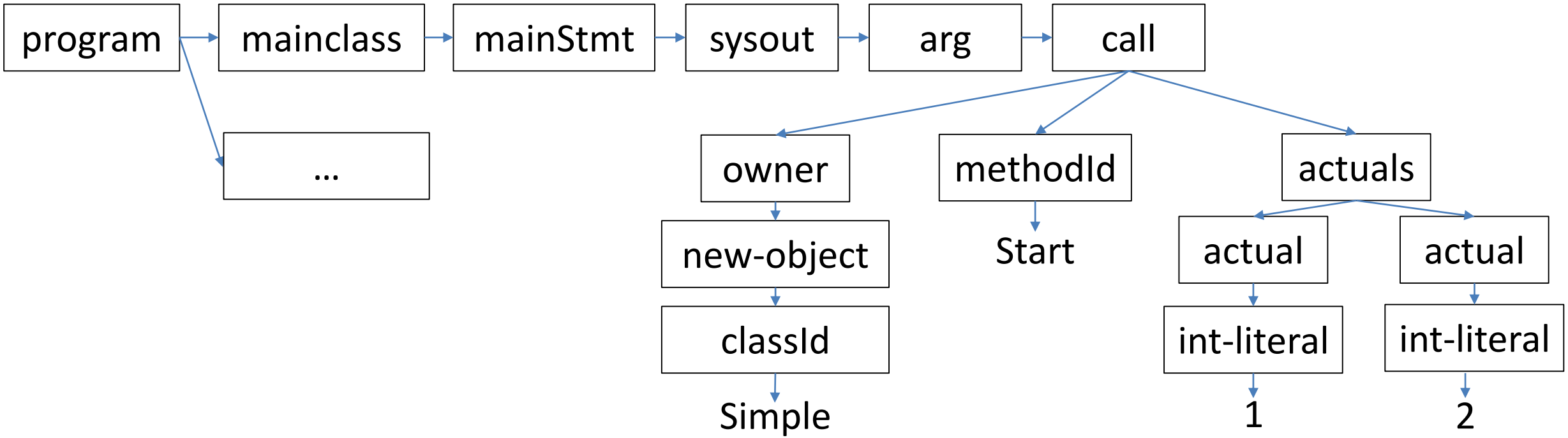
```
class Simple extends Trivial {
        public int Start(int a, int b) {
                int x;
                int y;

                x = a;
                y = b + 3;

                if (true) {
                        f = 0;
                } else {
                        f = 1;
                }

                return x + y + f;
        }
}
```

MiniJava Grammar

```
Goal ::= MainClass ( ClassDeclaration )* <EOF>
MainClass ::= "class" Identifier "{" "public" "static" "void" "main" "(" "String" "[" "]" Identifier ")" "{" Statement "}" "}"
ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{" ( VarDeclaration )* ( MethodDeclaration )* "}"
VarDeclaration ::= Type Identifier ";"
MethodDeclaration ::= "public" Type Identifier "(" ( Type Identifier ( "," Type Identifier )* )? ")" "{" ( VarDeclaration )* ( Statement )* "return" Expression ";" "}"
Type ::= "int" "[" "]"
       | "boolean"
       | "int"
       | Identifier
Statement ::= "{" ( Statement )* "}"
       | "if" "(" Expression ")" Statement "else" Statement
       | "while" "(" Expression ")" Statement
       | "System.out.println" "(" Expression ")" ";"
       | Identifier "=" Expression ";"
       | Identifier "[" Expression "]" "=" Expression ";"
Expression ::= Expression ( "&&" | "<" | "+" | "-" | "*" ) Expression
       | Expression "[" Expression "]"
       | Expression "." "length"
       | Expression "." Identifier "(" ( Expression ( "," Expression )* )? ")"
       | <INTEGER_LITERAL>
       | "true"
```

...

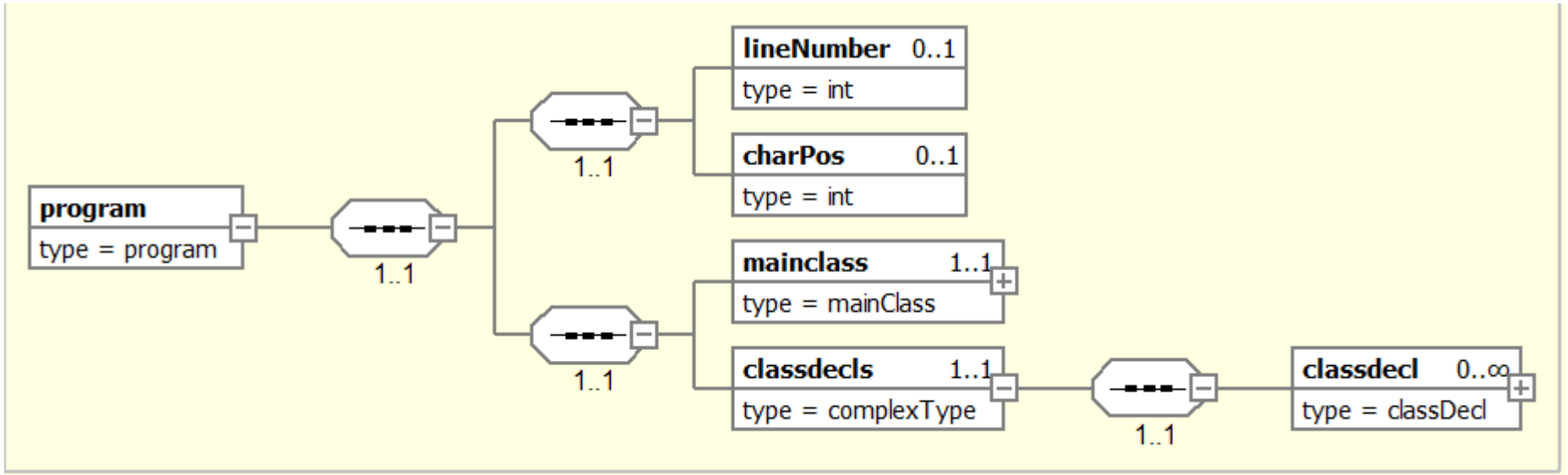
Simple Example AST (partial)



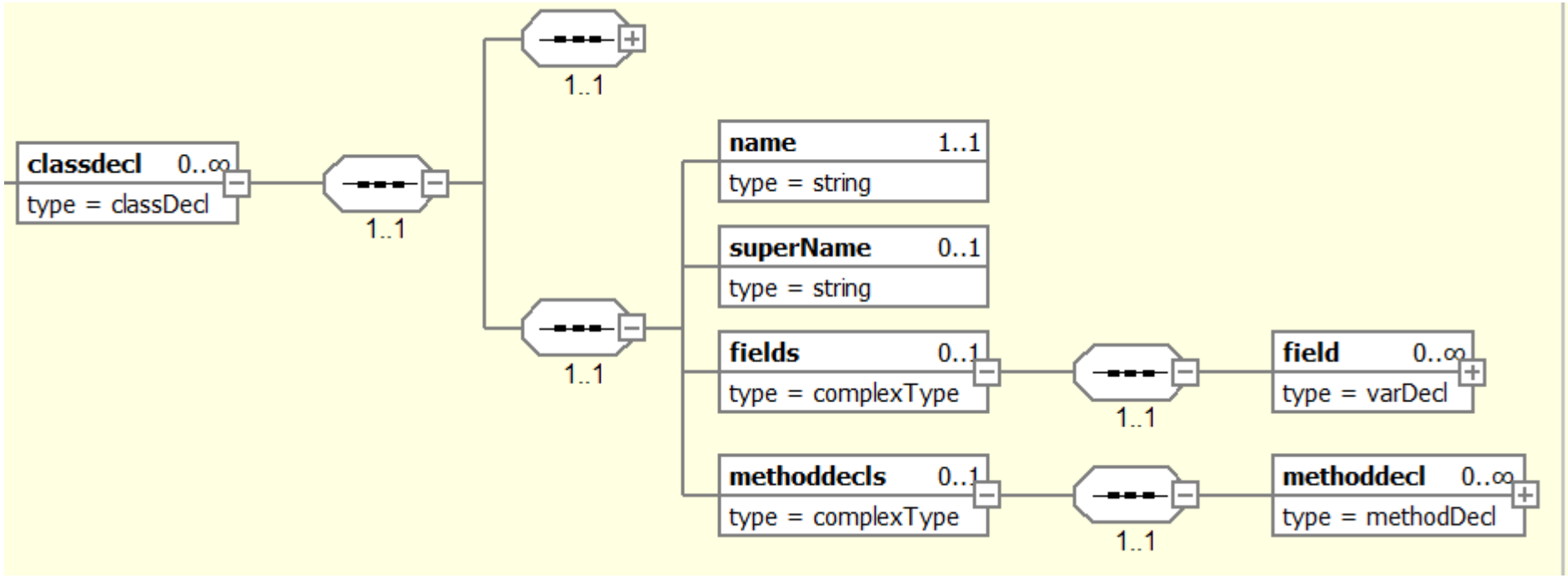

```
class Main {
    public static void main(String[] a) {
        System.out.println(new Simple().Start(1, 2));
    }
}
…
```

Writing MiniJava ASTs in XML

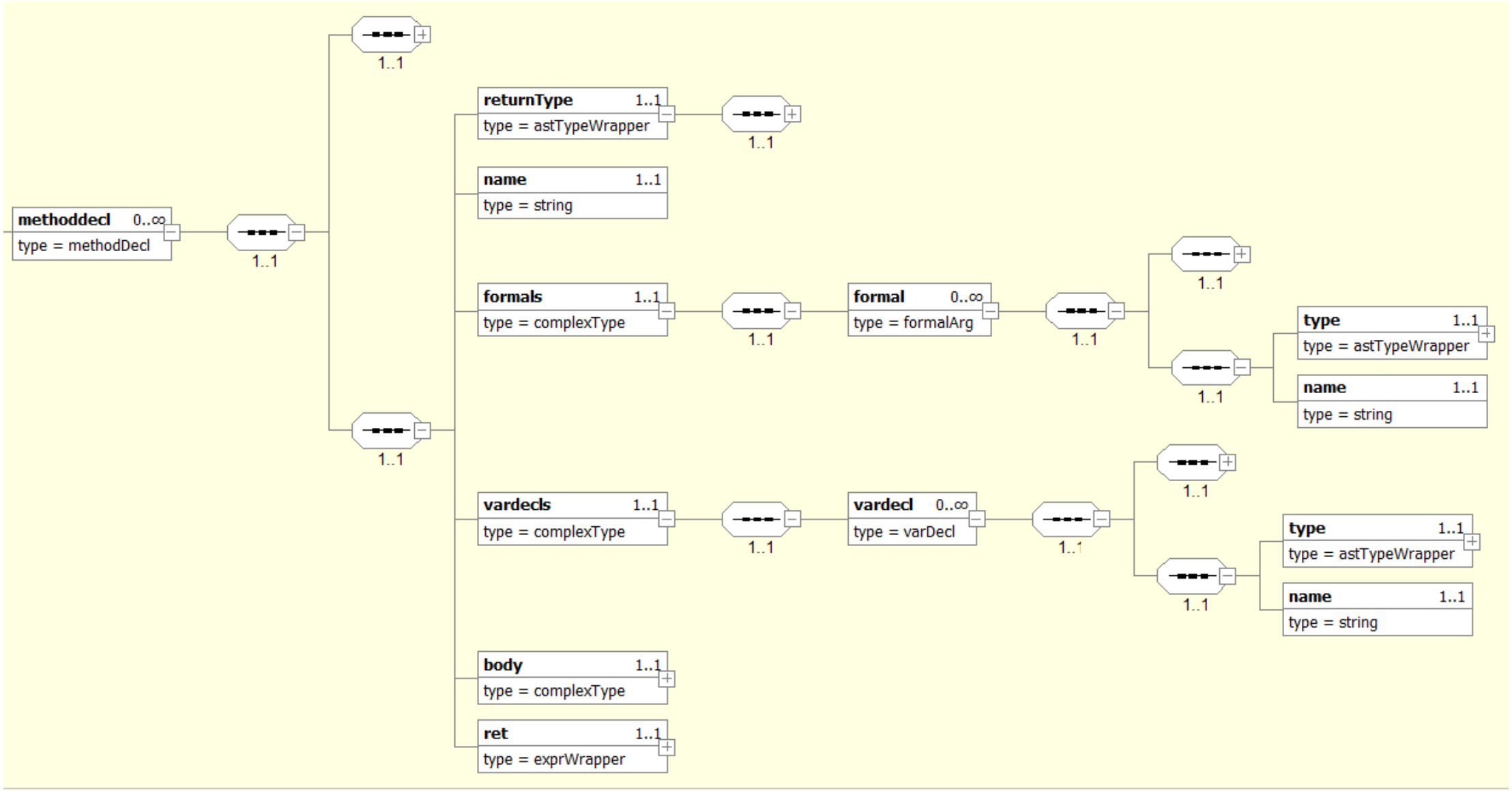
MiniJava AST



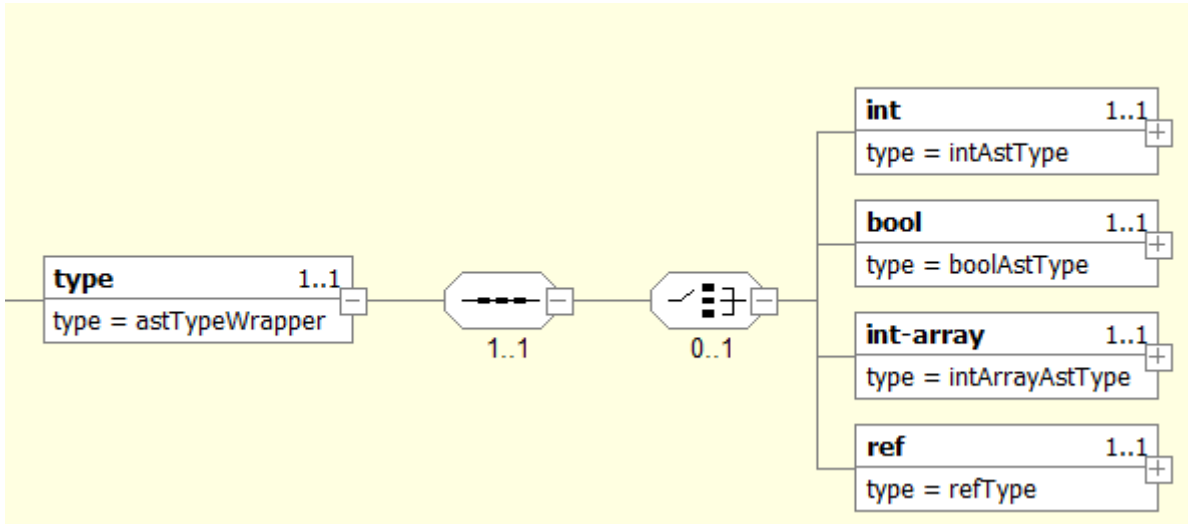
Class Declarations



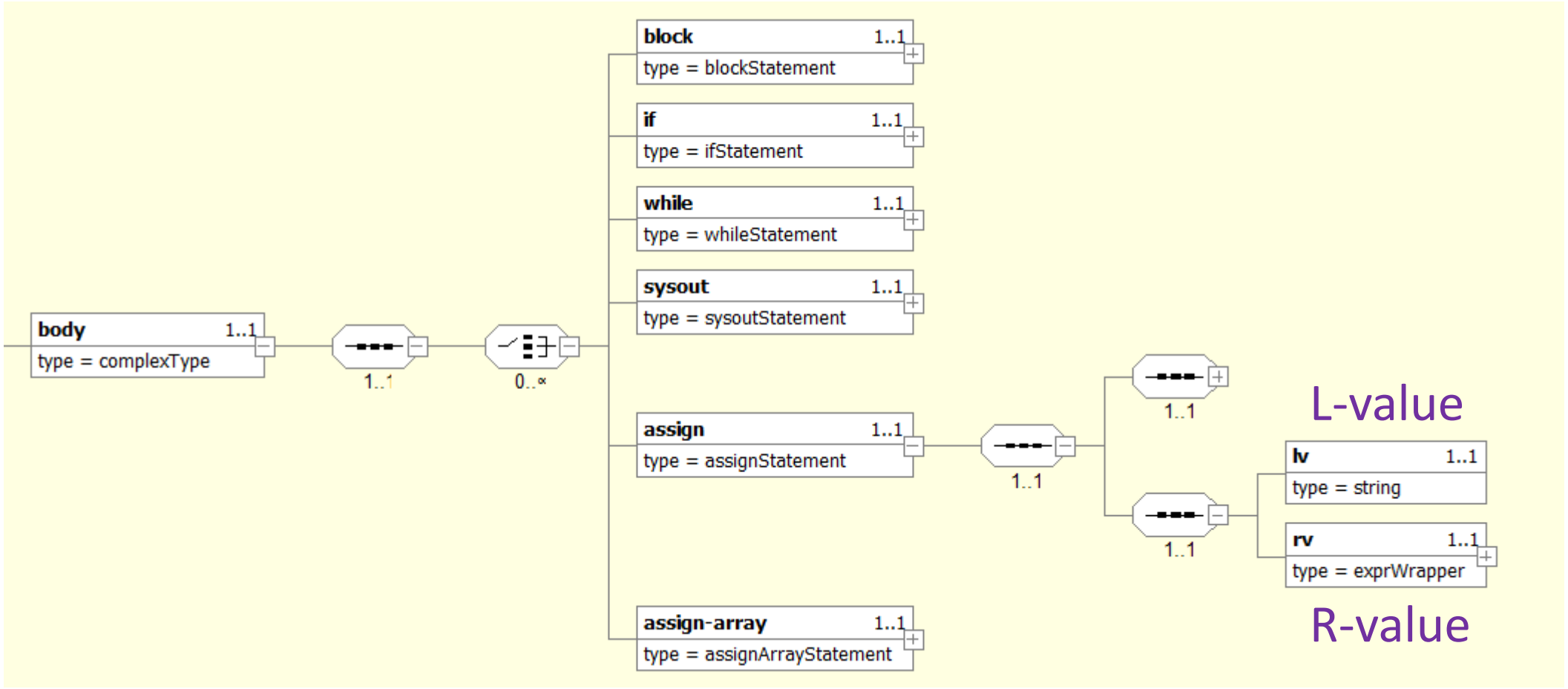
Method Declarations



Types


type
1..1
type = astTypeWrapper
1..1
0..1
int
1..1
type = intAstType
bool
1..1
type = boolAstType
int-array
1..1
type = intArrayAstType
ref
1..1
type = refType

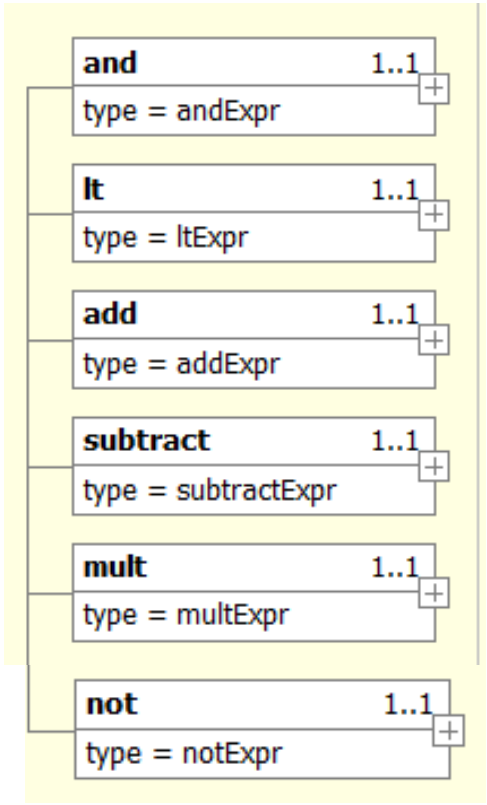
Statements

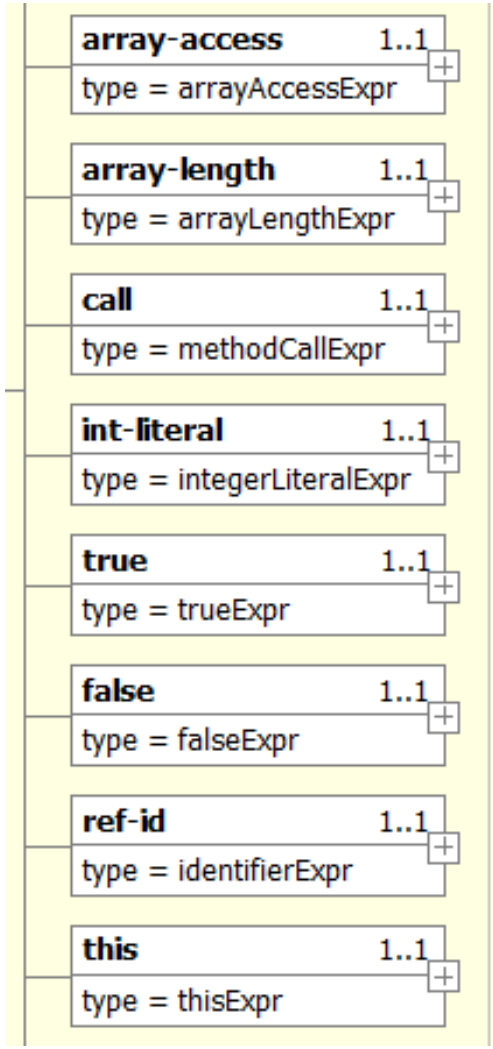

body
1..1
type = complexType
1..1
0..∞
block
1..1
type = blockStatement
if
1..1
type = ifStatement
while
1..1
type = whileStatement
sysout
1..1
type = sysoutStatement
assign
1..1
type = assignStatement
assign-array
1..1
type = assignArrayStatement
1..1
1..1
1..1
lv
1..1
type = string
rv
1..1
type = exprWrapper
L-value
R-value

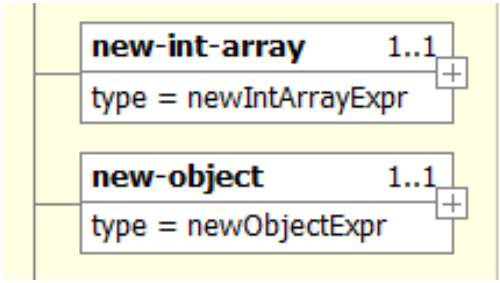
L-Values & R-Values

- `x = 3;`
- `x = y;`
- `x = new B().Start();`
- `new B().Start() = x;`
- `x.f = y;`

Expressions







Formal & Actual Parameters

- `public int Start(int a, int b)`
- `new Simple().Start(a, b)`

Writing AST XMLs

- Validate against the schema.

Summary

- MiniJava
- MiniJava AST
  - declarations
  - statements vs. expressions
  - formal vs. actual parameters
  - l-value vs. r-value
  - …
- XML representation of ASTs