**Simple Language with Local Variables**

This is a Java-like language that we use to illustrate basic concepts of scopes and type checking.

For simplicity, assume

* programs have only one class (called *World*)
* all class members are *static*.

The following is *abstract*, not *concrete* syntax

* some commas and semicolons are removed

But, for readabiility:

* keywords, quotes, and parantheses are kept to make the definitions clearer

In the text below we do not use quotes around keywords, usually use **bold** font instead.

program ::= "class" "World" "{" varDecl\* method\* "}"

method ::= varDecl "(" varDecl\* ")" "{" thing\* "return" expr "}"

varDecl ::= type ID

type ::= "int" | "boolean" | "void"

thing ::= varDecl | stmt

stmt ::= expr | if | while | block

if ::= "if" expr stmt "else" stmt

while ::= "while""("expr")" stmt

block ::= "{" thing\* "}"

expr ::= ID | expr "+" expr | expr "<=" expr

| assign | call | condExpr

assign ::= ID "=" expr

condExpr ::= expr "?" expr ":" expr

call ::= ID "(" expr\* ")"

Remark on language:

* blocks can be arbitrarily nested, and can contain new local variable declarations
* method call is an expression (resulting type is return type of method)
* assignment is an expression (resulting type is void, so ‘x=(y=z)’ is not correct according to types)