

## EBNF of Mini Java cfg

Program  $\Rightarrow$  (ClassDecl)\* eof

ClassDecl  $\Rightarrow$  class id { (Vis Access (Type id (; | MethodDecl)  
| void id MethodDecl))\* }

FieldDecl  $\Rightarrow$  ;

MethodDecl  $\Rightarrow$  (ParamList?) { Stmt\* }

Vis  $\Rightarrow$  ( public | private )?

Access  $\Rightarrow$  static?

Type  $\Rightarrow$  boolean | (int | id) (  $\epsilon$  | [ ] )

ParamList  $\Rightarrow$  Type id (, Type id)\*

ArgList  $\Rightarrow$  Expr (, Expr)\*

Ref  $\Rightarrow$  id | this | Ref. id

Stmt  $\Rightarrow$  { Stmt\* } | Type id = Expr ;  
| Ref ( = Expr ; | [ Expr ] = Expr ; | ( ArgList? ); )  
| return Expr? ;  
| if ( Expr ) Stmt ( else Stmt )?  
| while ( Expr ) Stmt

Expr  $\Rightarrow$  (  
Ref [ [ Expr ] | ( ArgList? ) ]  $\epsilon$  )  
| unop Expr  
| ( Expr ) | num | true | false  
| new ( id ( ( ) | [ Expr ] ) | int [ Expr ] )  
| ( binop Expr )\*

## Transformations made to the grammar

Class Decl  $\Rightarrow$  class *id* { (FieldDecl | MethodDecl)\* }

FieldDecl  $\Rightarrow$  Vis Access Type *id* ;

MethodDecl  $\Rightarrow$  Vis Access (Type | void) *id* (ParamList?) { Stmt\* }

Type  $\Rightarrow$  int | boolean | *id* | (int | *id*) [ ] left factor

Stmt  $\Rightarrow$  { Stmt\* } | Type *id* = Expr ;  
left factor | Ref = Expr ;  
 | Ref [ Expr ] = Expr ;  
 | Ref ( ArgList? ) ;  
 | return Expr? ;  
 | if ( Expr ) Stmt ( else Stmt ) ?  
 | while ( Expr ) Stmt

Expr  $\Rightarrow$  Ref left factor  
 | Ref [ Expr ]  
 | Ref ( ArgList? )  
 | unop Expr  
 | Expr binop Expr left recursion  
 | ( Expr )  
 | ... | | |

	num		True		false		left false
	new	(	id()		int [ Expr ]		id [ Expr ]