

Concrete Syntax	Abstract Syntax
$\langle \text{Program} \rangle ::= \text{prog IDENTIFIER } \langle \text{Block} \rangle \text{ gorp EOF}$	$\langle \text{Program} \rangle ::= \text{IDENTIFIER } \langle \text{Block} \rangle$
$\langle \text{Block} \rangle ::= (\langle \text{Declaration} \rangle ; \langle \text{Command} \rangle ;)^*$	$\langle \text{Block} \rangle ::= \langle \text{DecOrCommand} \rangle^*$
$\langle \text{Declaration} \rangle ::= \langle \text{Type} \rangle \text{ IDENTIFIER}$	$\langle \text{Declaration} \rangle ::= \langle \text{Type} \rangle \text{ IDENTIFIER}$
$\langle \text{Type} \rangle ::= \langle \text{SimpleType} \rangle \langle \text{CompoundType} \rangle$	
$\langle \text{SimpleType} \rangle ::= \text{int} \text{boolean} \text{string}$	$\langle \text{SimpleType} \rangle ::= \text{int} \text{boolean} \text{string}$ (use enum in Kind class)
$\langle \text{CompoundType} \rangle ::= \text{map} [\langle \text{SimpleType} \rangle , \langle \text{Type} \rangle]$	$\langle \text{CompoundType} \rangle ::= \langle \text{SimpleType} \rangle \langle \text{Type} \rangle$
$\langle \text{Command} \rangle ::= \langle \text{LValue} \rangle = \langle \text{Expression} \rangle$	$\langle \text{AssignExprCommand} \rangle ::= \langle \text{LValue} \rangle \langle \text{Expression} \rangle$
$ \langle \text{LValue} \rangle = \langle \text{PairList} \rangle$	$\langle \text{AssignPairListCommand} \rangle ::= \langle \text{LValue} \rangle \langle \text{PairList} \rangle$
$ \text{print } \langle \text{Expression} \rangle$	$\langle \text{PrintCommand} \rangle ::= \langle \text{Expression} \rangle$
$ \text{println } \langle \text{Expression} \rangle$	$\langle \text{PrintLnCommand} \rangle ::= \langle \text{Expression} \rangle$
$ \text{do } (\langle \text{Expression} \rangle) \langle \text{Block} \rangle \text{ od}$	$\langle \text{DoCommand} \rangle ::= \langle \text{Expression} \rangle \langle \text{Block} \rangle$
$\text{IDENTIFIER} [\text{do } \langle \text{LValue} \rangle : [\text{IDENTIFIER} , \langle \text{Block} \rangle \text{ od}$	$\langle \text{DoEachCommand} \rangle ::= \langle \text{LValue} \rangle \text{ Identifier Identifier } \langle \text{Block} \rangle$
$ \text{if } (\langle \text{Expression} \rangle) \langle \text{Block} \rangle \text{ fi}$	$\langle \text{IfCommand} \rangle ::= \langle \text{Expression} \rangle \langle \text{Block} \rangle$
$ \text{if } (\langle \text{Expression} \rangle) \langle \text{Block} \rangle$	$\langle \text{IfElseCommand} \rangle ::= \langle \text{Expression} \rangle \langle \text{Block} \rangle$
$\text{else } \langle \text{Block} \rangle \text{ fi}$	$\langle \text{Block} \rangle$
$ \epsilon$	
$\langle \text{LValue} \rangle ::= \text{IDENTIFIER}$	$\langle \text{SimpleLValue} \rangle ::= \text{IDENTIFIER}$
$\langle \text{LValue} \rangle ::= \text{IDENTIFIER} [\langle \text{Expression} \rangle]$	$\langle \text{ExprLValue} \rangle ::= \text{IDENTIFIER } \langle \text{Expression} \rangle$
$\langle \text{Pair} \rangle ::= [\langle \text{Expression} \rangle , \langle \text{Expression} \rangle]$	$\langle \text{Pair} \rangle ::= \langle \text{Expression} \rangle \langle \text{Expression} \rangle$
$\langle \text{PairList} \rangle ::= \{ \langle \text{Pair} \rangle (, \langle \text{Pair} \rangle)^* \} \{ \}$	$\langle \text{PairList} \rangle ::= \langle \text{Pair} \rangle^*$
$\langle \text{Expression} \rangle ::= \langle \text{Term} \rangle (\langle \text{RelOp} \rangle \langle \text{Term} \rangle)^*$	$\langle \text{BinaryOpExpression} \rangle ::= \langle \text{Expression} \rangle \text{ op } \langle \text{Expression} \rangle$ (use enum in Kind class for Op)
$\langle \text{Term} \rangle ::= \langle \text{Elem} \rangle (\langle \text{WeakOp} \rangle \langle \text{Elem} \rangle)^*$	
$\langle \text{Elem} \rangle ::= \langle \text{Factor} \rangle (\langle \text{StrongOp} \rangle \langle \text{Factor} \rangle)^*$	
$\langle \text{Factor} \rangle ::= \langle \text{LValue} \rangle$	$\langle \text{LValueExpression} \rangle ::= \langle \text{LValue} \rangle$
$\langle \text{Factor} \rangle ::= \text{INTEGER_LITERAL}$	$\langle \text{IntegerLiteralExpression} \rangle ::= \text{INTEGER_LITERAL}$
$\langle \text{Factor} \rangle ::= \text{BOOLEAN_LITERAL}$	$\langle \text{BooleanLiteralExpression} \rangle ::= \text{BOOLEAN_LITERAL}$
$\langle \text{Factor} \rangle ::= \text{STRING_LITERAL}$	$\langle \text{IntegerLiteralExpression} \rangle ::= \text{STRING_LITERAL}$
$\langle \text{Factor} \rangle ::= (\langle \text{Expression} \rangle)$	
$\langle \text{Factor} \rangle ::= ! \langle \text{Factor} \rangle$	$\langle \text{UnaryOpExpression} \rangle ::= \text{op } \langle \text{Expression} \rangle$ (use enum in Kind class for Op)
$\langle \text{Factor} \rangle ::= - \langle \text{Factor} \rangle$	$\langle \text{UnaryOpExpression} \rangle ::= \text{op } \langle \text{Expression} \rangle$ (use enum in Kind class for Op)

$\langle \text{RelOp} \rangle ::= \text{OR} \mid \text{AND} \mid \text{EQUALS} \mid$ $\text{NOT_EQUALS} \mid \text{LESS_THAN} \mid$ $\text{GREATER_THAN} \mid \text{AT_MOST} \mid \text{AT_LEAST}$ $\langle \text{WeakOp} \rangle ::= \text{PLUS} \mid \text{MINUS}$ $\langle \text{StrongOp} \rangle ::= \text{TIMES} \mid \text{DIVIDE}$	(use enum in Kind class for Op)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------

Recall that symbols with multiple rules map to an abstract class, instances of the rule are subclasses. Example:

```

abstract class Type{}
class SimpleType extends Type{...}
class CompoundType extends Type{...}

```