

# Languages-beta: SIMPLE-3-Statements \*

The PPlanCompS Project

SIMPLE-3-Statements.cbs | PLAIN | PRETTY

---

Language "SIMPLE"

## 3 Statements

*Syntax*  $Block : block ::= \{ stmts? \}$   
 $Stmts : stmts ::= stmt stmts?$   
 $Stmt : stmt ::= imp-stmt | vars-decl$   
 $ImpStmt : imp-stmt ::= block$   
|  $exp \ ;$   
|  $\text{'if' ' (' exp ') ' block ('else' block)?}$   
|  $\text{'while' ' (' exp ') ' block}$   
|  $\text{'for' ' (' stmt exp ';' exp ') ' block}$   
|  $\text{'print' ' (' exps ') ' ';'}$   
|  $\text{'return' exp? ';'}$   
|  $\text{'try' block 'catch' ' (' id ') ' block}$   
|  $\text{'throw' exp ';'}$

*Rule*  $\llbracket \text{'if' ' (' Exp ') ' Block} \rrbracket : stmt =$   
 $\llbracket \text{'if' ' (' Exp ') ' Block 'else' '{' '}' } \rrbracket$   
*Rule*  $\llbracket \text{'for' ' (' Stmt Exp_1 ';' Exp_2 ') '}$   
 $\text{'{' Stmt '}' } \rrbracket : stmt =$   
 $\llbracket \text{'{' Stmt}$   
 $\text{'while' ' (' Exp_1 ') '}$   
 $\text{'{' '{' Stmt '}' Exp_2 ';' '}'}$   
 $\text{'}' } \rrbracket$

---

\*Suggestions for improvement: [plancomps@gmail.com](mailto:plancomps@gmail.com).  
Reports of issues: <https://github.com/plancomps/CBS-beta/issues>.

Semantics  $\text{exec}[\_ : \text{stmts}] : \Rightarrow \text{null-type}$

Rule  $\text{exec}[\text{'{' '}'}] = \text{null}$

Rule  $\text{exec}[\text{'{' Stmt '}'}] = \text{exec}[\text{Stmt}]$

Rule  $\text{exec}[\text{ImpStmt Stmt}] =$   
 $\text{sequential}(\text{exec}[\text{ImpStmt}], \text{exec}[\text{Stmt}])$

Rule  $\text{exec}[\text{VarsDecl Stmt}] =$   
 $\text{scope}(\text{declare}[\text{VarsDecl}], \text{exec}[\text{Stmt}])$

Rule  $\text{exec}[\text{VarsDecl}] = \text{effect}(\text{declare}[\text{VarsDecl}])$

Rule  $\text{exec}[\text{Exp ';' }] = \text{effect}(\text{rval}[\text{Exp}])$

Rule  $\text{exec}[\text{'if' '(' Exp ')' Block<sub>1</sub> 'else' Block<sub>2</sub>}] =$   
 $\text{if-else}(\text{rval}[\text{Exp}], \text{exec}[\text{Block<sub>1</sub>}], \text{exec}[\text{Block<sub>2</sub>}] )$

Rule  $\text{exec}[\text{'while' '(' Exp ')' Block}] = \text{while}(\text{rval}[\text{Exp}], \text{exec}[\text{Block}])$

Rule  $\text{exec}[\text{'print' '(' Exps ')' ';' }] = \text{print}(\text{rvals}[\text{Exps}])$

Rule  $\text{exec}[\text{'return' Exp ';' }] = \text{return}(\text{rval}[\text{Exp}])$

Rule  $\text{exec}[\text{'return' ';' }] = \text{return}(\text{null})$

Rule  $\text{exec}[\text{'try' Block<sub>1</sub> 'catch' '(' Id ')' Block<sub>2</sub>}] =$   
 $\text{handle-thrown}(\text{exec}[\text{Block<sub>1</sub>}],$   
 $\text{scope}(\text{bind}(\text{id}[\text{Id}], \text{allocate-initialised-variable}(\text{values, given})),$   
 $\text{exec}[\text{Block<sub>2</sub>}] ) )$

Rule  $\text{exec}[\text{'throw' Exp ';' }] = \text{throw}(\text{rval}[\text{Exp}])$