Original Grammar

Program ::= *Decl*+

Decl ::= *VariableDecl* | *FunctionDecl* | *ConstDecl* | *ClassDecl* | *IntefaceDecl*

VariableDecl ::= *Variable* **;**

Variable ::= *Type* **ident**

ConstDecl ::= **static** *ConstType* **ident** *;*

ConstType ::= **int** | **double** | **boolean** | **string**

Type ::= **int** | **double** | **boolean** | **string** | **ident** | *Type***[ ]**

FunctionDecl ::= *Type* **ident (** *Formals* **)** *StmtBlock* | **void ident (** *Formals* **)** *StmtBlock*

Formals ::= *Variable* **,** *Formals* | *Variable*

ClassDecl ::= **class ident** < **extends ident**> < **implements ident**+ **,** > **{** *Field** **}**

Field ::= *VariableDecl* | *FunctionDecl* | *ConstDecl*

InterfaceDecl ::= **interface ident {** *Prototype** **}**

Prototype ::= *Type* **ident (** *Formals* **) ;** | **void ident (** *Formals* **) ;**

StmtBlock ::= **{** *VariableDecl** *ConstDecl** *Stmt** **}**

Stmt ::= < *Expr* > **;** | *IfStmt* | *WhileStmt* | *ForStmt* | *BreakStmt* | *ReturnStmt* | *PrintStmt*
| *StmtBlock*

IfStmt ::= **if (** *Expr* **)** *Stmt* < **else** *Stmt* >

WhileStmt ::= **while (** *Expr* **)** *Stmt*

ForStmt ::= **for (** *Expr* **;** *Expr* **;** *Expr* **)** *Stmt*

ReturnStmt ::= **return** *Expr* **;**

BreakStmt ::= **break ;**

PrintStmt ::= **System.out.println (** *Expr*+ **,) ;**

Expr ::= *LValue* **=** *Expr* | *Constant* | *LValue* | **this** | **(** *Expr* **)** | *Expr* **-** *Expr* | *Expr* **/** *Expr*
| *Expr* **%** *Expr* | **-** *Expr* | *Expr* **>** *Expr* | *Expr* **>=** *Expr* | *Expr* **!=** *Expr* | *Expr* **||** *Expr*
| **!** *Expr* | **New (ident)**

LValue ::= **ident** | *Expr* **. ident**

Constant ::= **intConstant** | **doubleConstant** | **booleanConstant** | **stringConstant** | **null**

Expanded Grammar

Init → *Program*

1. *Program* → *Decl Program*
2. *Program* → *Decl*
3. *DeclAdditional* → *Type* **ident ;**
4. *DeclAdditional* → *FuncProtoInit* **ident (** *Formals* **)** *StmtBlock*
5. *DeclAdditional* → **static** *ConstType* **ident ;**
6. *Decl* → *DeclAdditional*
7. *Decl* → **class ident** *Extends Implements* **{** *Field* **}**
8. *Decl* → **interface ident {** *Prototype* **}**
9. *ConstType* → **int**
10. *ConstType* → **double**
11. *ConstType* → **boolean**
12. *ConstType* → **string**
13. *Type* → *ConstType*
14. *Type* → **ident**
15. *Type* → *Type* **[]**
16. *FuncProtoInit* → *Type*
17. *FuncProtoInit* → **void**
18. *Formals* → *Type* **ident ,** *Formals*
19. *Formals* → *Type* **ident**
20. *Extends* → *extends* **ident**
21. *Extends* → *ε*
22. *Implements* → *Implements* **ident** *ImplementsIdentPlus*
23. *Implements* → *ε*
24. *ImplementsIdentPlus* → **, ident** *ImplementsIdentPlus*
25. *ImplementsIdentPlus* → *ε*
26. *Field* → *DeclAdditional Field*
27. *Field* → *ε*
28. *Prototype* → *FuncProtoInit* **ident (** *Formals* **) ;** *Prototype*
29. *Prototype* → *ε*
30. *StmtBlock* → **{** *VariableDeclStar ConstDeclStar StmtStar* **}**
31. *VariableDeclStar* → *Type* **ident ;** *VariableDeclStar*
32. *VariableDeclStar* → *ε*
33. *ConstDeclStar* → **static** *ConstType* **ident ;** *ConstDeclStar*
34. *ConstDeclStar* → *ε*
35. *StmtStar* → *Stmt StmtStar*
36. *StmtStar* → *ε*
37. *Stmt* → *Expr* **;**
38. *Stmt* → **;**
39. *Stmt* → **if (** *Expr* **)** *Stmt ElseStmt*
40. *Stmt* → **while (** *Expr* **)** *Stmt*
41. *Stmt* → **for (** *Expr* **;** *Expr* **;** *Expr* **)** *Stmt*
42. *Stmt* → **break ;**
43. *Stmt* → **return** *Expr* **;**
44. *Stmt* → **System . out . println (** *Expr PrintStmtExpr* **) ;**
45. *Stmt* → *StmtBlock*
46. *ElseStmt* → **else** *Stmt*
47. *ElseStmt* → *ε*
48. *PrintStmtExpr* → **,** *Expr PrintStmtExpr*

49. *PrintStmtExpr* → *ε*
50. *Expr* → **ident** *Access* **=** *ExprSubLevel1*
51. *Expr* → *ExprSubLevel1*
52. *ExprSubLevel1* → *ExprSubLevel1* **||** *ExprSubLevel2*
53. *ExprSubLevel1* → *ExprSubLevel2*
54. *ExprSubLevel2* → *ExprSubLevel2* **!=** *ExprSubLevel3*
55. *ExprSubLevel2* → *ExprSubLevel3*
56. *ExprSubLevel3* → *ExprSubLevel3* **>** *ExprSubLevel4*
57. *ExprSubLevel3* → *ExprSubLevel3* **>=** *ExprSubLevel4*
58. *ExprSubLevel3* → *ExprSubLevel4*
59. *ExprSubLevel4* → *ExprSubLevel5* **-** *ExprSubLevel6*
60. *ExprSubLevel4* → *ExprSubLevel5*
61. *ExprSubLevel5* → *ExprSubLevel5 I ExprSubLevel6*
62. *ExprSubLevel5* → *ExprSubLevel5* **%** *ExprSubLevel6*
63. *ExprSubLevel5* → *ExprSubLevel6*
64. *ExprSubLevel6* → **New (ident)**
65. *ExprSubLevel6* → *ExprSubLevel7*
66. *ExprSubLevel7* → **-** *ExprSubLevel8*
67. *ExprSubLevel7* → **!** *ExprSubLevel8*
68. *ExprSubLevel7* → *ExprSubLevel8*
69. *ExprSubLevel8* → **(** *Expr* **)**
70. *ExprSubLevel8* → **this**
71. *ExprSubLevel8* → **intConstant**
72. *ExprSubLevel8* → **doubleConstant**
73. *ExprSubLevel8* → **booleanConstant**
74. *ExprSubLevel8* → **stringConstant**
75. *ExprSubLevel8* → **null**
76. *ExprSubLevel8* → **ident** *Access*
77. *Access* → **.** **ident** *Access*
78. *Access* → *ε*

NOTE: Bold text are the terminals of the grammar.

First and Follow

Non Terminal	First	Follow
Init	static,class,interface,int,double,boolean,string,ident,void	$
Program	static,class,interface,int,double,boolean,string,ident,void	$
DeclAdditional	static,int,double,boolean,string,ident,void	static,class,interface,int,double,boolean,string,ident,void,$,}
Decl	static,class,interface,int,double,boolean,string,ident,void	static,class,interface,int,double,boolean,string,ident,void,$
ConstType	int,double,boolean,string	ident,[]
Type	int,double,boolean,string,ident	ident,[]
FuncProtoInit	int,double,boolean,string,ident,void	ident
Formals	int,double,boolean,string,ident	)
Extends	extends,*ε*	implements,static,class,interface,int,double,boolean,string,ident,void,$,{
Implements	implements,*ε*	{
ImplementsIdentPlus	,,*ε*	{
Field	static,*ε*,int,double,boolean,string,ident,void	}
Prototype	int,double,boolean,string,ident,void,*ε*	}
StmtBlock	{	},;,if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,static,class,interface,int,double,boolean,string,void,$
VariableDeclStar	int,double,boolean,string,ident,*ε*	static,;,if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,},else,class,interface,int,double,boolean,string,void,$
ConstDeclStar	static,*ε*	;,if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,},else,static,class,interface,int,double,boolean,string,void,$
StmtStar	*ε*,;,if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null	}
Stmt	;,if,while,for,break,return,System,{,ident,New,-	},;,if,while,for,break,return,System,{,

	!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null	ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else
ElseStmt	else,*ε*	},;,if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else
PrintStmtExpr	,,*ε*	)
Expr	ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else
ExprSubLevel1	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||
ExprSubLevel2	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=
ExprSubLevel3	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=
ExprSubLevel4	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=,/,%
ExprSubLevel5	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=,/,%
ExprSubLevel6	New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=,/,%
ExprSubLevel7	-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=,/,%
ExprSubLevel8	(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,ident	;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,stringConstant,null,else,||,!=,>,>=,/,%
Access	.,*ε*	=,;,),,,},if,while,for,break,return,System,{,ident,New,-,!,(,this,intConstant,doubleConstant,booleanConstant,st

		ringConstant,null,else,||,!=,>,>=,/,%

Parsing Table

In the parsing table file.