1. Program →Start-Symbols ClassDeclaration End-Symbols.

2. Start-Symbols →@| ^

3. End-Symbols→$ |#

4. ClassDeclaration→ Type ID{ Class_Implementation} | Type ID Infer { Class_Implementation}

5. Class_Implementation→ Variable_Decl Class_Implementation| Method_Decl Class_Implementation | Comment Class_Implementation | require_command Class_Implementation| Func _Call Class_Implementation

|em

6. Method_Decl→ Func Decl ;| Func Decl { Variable_Decl Statements }

7. Func Decl →Type ID (ParameterList)

8. Type → Ipok |Sipok |Craf |Sequence |Ipokf |Sipokf |Valueless |Rational

9. ParameterList →em| None | Non-Empty List

10. Non-Empty List→ Type ID Non-Empty List *

Non-Empty List* → Type ID Non-Empty List* | e

11. Variable_Decl→ em | Type ID_List ; Variable_Decl | Type ID_List [ID] ; Variable_Decl

12. ID_List →ID ID_List *

ID_List* → ID ID_List * | e

13. Statements→em | Statement Statements

14. Statement→Assignment | If _Statement | However _Statement |

when_Statement | Respondwith _ Statement | Endthis

_Statement|Scanvalur (ID); | Print (Expression); |

15. Assignment→ Variable_Decl = Expression;

16. Func _Call → ID (Argument_List) ;

17. Argument_List →em | NonEmpty_Argument_List

18. NonEmpty_Argument_List →Expression NonEmpty_Argument_List *

NonEmpty_Argument_List ,* → Expression NonEmpty_Argument_List * | e

19. Block Statements→{ statements }

20. If _Statement→ if (Condition _Expression) Block Statements | if

(Condition _Expression) Block Statements else Block Statements

21. Condition _Expression→ Condition |Condition Condition _Op Condition

22. Condition _Op → && | ||

23. Condition→ Expression Comparison _Op Expression

24. Comparison _Op → == | != | > | >= | < | <=

25. However _Statement → However (Condition _Expression) Block Statements

26. when _Statement → when (expression ; expression ; expression) Block Statements

27. Respondwith _Statement→ Respondwith Expression ; | return ID ;

28. Endthis _Statement→ Endthis;

29. Expression → Term Expression *

Expression* → Add_Op Term Expression* | e

30. Add_Op → + | -

31. Term→Factor Term *

Term * → Mul_Op Factor Term * |e

32. Mul_Op→* | /

33. Factor→ ID| Number

34. Comment →</ STR /> | ***STR

35. Require_command →Require(F_name.txt);

36. F_name →STR

First calculation :

First(Program) → @, ^

First(Start-Symbols) → @, ^

First(End-Symbols)→ $,#

First(ClassDeclaration)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(Class_Implementation)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require , e

First(Method_Decl)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(Func Decl)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(Type)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(ParameterList)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , e ,None

First(Non-Empty List)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(Non-Empty List*)→ , , e

First(Variable_Decl)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , e

First(ID_List)→ ID

First(ID_List *)→ e

First(Statements)→ e , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when_Statement , Respondwith, Endthis , Scanvalur, Print,

First(Statement)→ e , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If , However ,when, Respondwith , Endthis , Scanvalur, Print,

First(Assignment)→ e , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational

First(Func _Call)→ ID

First(Argument_List)→ em , ID, Number

First(NonEmpty_Argument_List)→ ID, Number

First(NonEmpty_Argument_List*)→ ID, Number ,e

First(Block Statements)→ {

First(If _Statement)→ if

First(Condition _Expression)→ ID , Number

First(Condition _Op)→ && , ||

First(Condition)→ ID , Number

First(Comparison _Op)→ == , != , > , >= , < , <=

First(However _Statement)→ However

First(when _Statement)→ when

First(Respondwith _Statement)→ Respondwith , return

First(Endthis)→ Endthis

First(Expression)→ ID , Number

First(Expression*)→ + , - , e

First(ADD_op)→ + , -

First(Term)→ ID , Number

First(Term *)→ * , / ,e

First(Mul_Op)→ * , /

First(Factor)→ ID , Number

First(Comment) →</ STR /> , ***STR

First(Require_command) → Require

First(F_name) → STR

Follow calculation :

Follow (Program) → $

Follow (Start-Symbols) → }

Follow (End-Symbols) → $

Follow (ClassDeclaration) → $, #

Follow (Class_Implementation) → First(Class_Implementation)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require

Follow (Method_Decl) → ; , {

Follow (Func Decl) → (

Follow (type) → ID

Follow (ParameterList) →)

Follow (Non-Empty List) →)

Follow (Non-Empty List*) →)

Follow (. Variable_Decl) → { , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require = ,If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (. ID_List) → ; , [

Follow (. ID_List*) → ; , [

Follow (. Statements) → } ,

Follow (. Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (Assignment) → } Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (Func _Call) → First(Class_Implementation)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require ,

Follow (Argument_List) →)

Follow (NonEmpty_Argument_List) →)

Follow (NonEmpty_Argument_List*) →)

Follow (Block Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print, else ,

Follow (IF_statment) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when_ , Respondwith, Endthis , Scanvalur, Print,

Follow (Condition _Expression) →)

Follow (Condition _Op) → ID , Number

Follow (Condition) → && , || ,)

Follow (Comparison _Op) → ID , Number

Follow (However_ Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (When_ Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (Respondwith _Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when, Respondwith, Endthis , Scanvalur, Print,

Follow (Endthis _Statement) → } , Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , If, However ,when_ , Respondwith, Endthis , Scanvalur, Print,

Follow (Expression) → } ;) == , != , > , >= , < , <= ID , Number

Follow (Expression*) → } ;) == , != , > , >= , < , <= ID , Number

Follow (Add_Op) → ID , Number

Follow (Term) → + , -

Follow (Term*) → + , -

Follow (Mul_Op) → ID , Number

Follow (comment)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require ,

Follow (. Require_command)→ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational , ID ,</ STR /> , ***STR , Require ,

Follow (F_name)→ .