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)→ .