hw2

1.

extend BNF :

ClassDecl -> class id { VarDecl\* MethodDecl\* }

VarDecl -> Type id (= Exp)? (, id (= Exp)? )\* ;

MethodDecl -> public Type id “(“ FormalList? “)” { Statement\* }

FormalList -> Type id ( , Type id )\*

basic BNF:

1. ClassDecl -> class id {VarList MethodList}

2. VarList -> VarDecl VarList

3. | ℇ

4. MethodList -> MethodDecl MethodList

5. | ℇ

6. VarDecl -> VarType VarInit MoreVar ;

7. VarType -> Type id

8. VarInit -> = Exp

9. | ℇ

10. MoreVar -> , id VarInit MoreVar

11. | ℇ

12. MethodDecl -> public Type id “(” FormalList “)” { StmtList }

13. FormalList -> VarType TypeList

14. | ℇ

15. TypeList -> , VarType TypeList

16. | ℇ

17. StmtList -> Statement StmtList

18. | ℇ

2.

1. S -> L = R ;

2. S -> R ;

3. L -> ID

4. L -> \*R

5. R -> L

(1)

FIRST[1] = {ID, \*}

FIRST[2] = {ID, \*}

FIRST[3] = {ID}

FIRST[4] = {\*}

FIRST[5] = {ID, \*}

FIRST(S) = {ID, \*}

FIRST(L) = {ID, \*}

FIRST(R) = {ID, \*}

(2)

FOLLOW(S) = {$}

FOLLOW(L) = {=, ;}

FOLLOW(R) = {;, =}

(3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | = | ID | \* | ; | $ |
| S |  | 1, 2 | 1, 2 |  |  |
| L |  | 3 | 4 |  |  |
| R |  | 5 | 5 |  |  |

3.

grammar:

1. VarDecl -> Type VarList ;

2. VarList -> VarItem

3. VarList -> VarList , VarItem

4. VarItem -> ID

5. VarItem -> ID = num

3.1)

a) FIRST[1] = {Type}

FIRST[2] = {ID}

FIRST[3] = {ID}

FIRST[4] = {ID}

FIRST[5] = {ID}

b) FIRST(VarDecl) = {Type}

FIRST(VarList) = {ID}

FIRST(VarItem)= {ID}

c) FOLLOW(VarDecl) = {$}

FOLLOW(VarList) = {;, “,”}

FOLLOW(VarItem) = {;, “,”}

3.2)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Type | ID | , | = | num | ; | $ |
| VarDecl | 1 |  |  |  |  |  |  |
| VarList |  | 2, 3 |  |  |  |  |  |
| VarItem |  | 4, 5 |  |  |  |  |  |

4.

grammar:

1. VarDecl -> Type VarList ;

2. VarList -> VarItem

3. VarList -> VarList , VarItem

4. VarItem -> ID

5. VarItem -> ID = num

4.1)

1. VarDecl -> Type VarList;

//intermedia step: (extended BNF: VarList -> VarItem (, VarItem)\*)

2. VarList -> VarItem ItemList

3. ItemList -> , VarItem ItemList

4. | ℇ

5. VarItem -> ID NumItem

6. NumItem -> = num

7. | ℇ

4.2)

a) FIRST[1] = {Type}

FIRST[2] = {ID}

FIRST[3] = {,}

FIRST[4] = {}

FIRST[5] = {ID}

FIRST[6] = {=}

FIRST[7] = {}

b)

FIRST(VarDecl) = {Type}

FIRST(VarList) = {ID}

FIRST(VarItem) = {ID}

FIRST(ItemList) = {,}

FIRST(NumItem) = {=}

c)

FOLLOW(VarDecl) = {$}

FOLLOW(VarList) = {;}

FOLLOW(VarItem) = {“,”, ;}

FOLLOW(ItemList) = {;}

FOLLOW(NumItem) = {“,”, ;}

4.3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Type | ID | ; | , | = | $ |
| VarDecl | 1 |  |  |  |  |  |
| VarList |  | 2 |  |  |  |  |
| VarItem |  | 5 |  |  |  |  |
| ItemList |  |  | 4 | 3 |  |  |
| NumItem |  |  | 7 | 7 | 6 |  |

4.4) Type ID = num, ID ;

VarDecl ->(rule 1) Type VarList;

-> (rule 2) Type VarItem ItemList;

-> (rule 5) Type ID NumItem ItemList;

-> (rule 6) Type ID = num ItemList;

-> (rule 3) Type ID = num , VarItem ItemList;

-> (rule 5) Type ID = num , ID NumItem ItemList;

-> (rule 7) Type ID = num , ID ItemList;

-> (rule 4) Type ID = num , ID ;

5.

grammar:

1. Term → Term \* Value

2. | Term / Value

3. | Term % Value

4. | Value

5. Value → ! Factor

6. | Factor

equivalent LL(1):

// intermediate steps: Term -> TermHead (TermTail)\*

// TermHead -> Value

// TermTail -> \* Value

// | / Value

// | % Value

1. Term -> Value TermTail

2. TermTail -> \* Value TermTail

3. | / Value TermTail

4. | % Value TermTail

5. | ℇ

6. Value -> ! Factor

7. | Factor

LL(1) Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ! | Factor | \* | / | % | $ |
| Term | 1 | 1 |  |  |  |  |
| TermTail |  |  | 2 | 3 | 4 | 5 |
| Value | 6 | 7 |  |  |  |  |