Context Free Grammar for SCOOP

Prd #   Productions

1   <program> → <cls_list> $$

2   <cls_list> → <cls_list> <class>

3   <cls_list> → ∈

4   <class> → <st_dec_list> class cls_id { <cls_body> }   S

5   <st_dec_list> → [state (<st_list>)]   S

6   <st_dec_list> → ∈   S

7   <st_list> → <st_list> , <st_id>   S

8   <st_list> → <st_id>   S

9   <st_id> → st_id   S

10   <cls_body> → <var_dec_list> < st_binding> <cls_m_list> <c_s_m_list>   S

11   <var_dec_list>→<var_dec_list> <var_dec> ;

12   <var_dec_list> → ∈

13   <var_dec>→<vd_start><vd_rest>

14   <vd_start> → <pub-prv><dt> id

15   <dt> → int

16   <dt> → string

17   <dt> → cl_id

18   <st_binding> → statebinding { <b_list> }   S

19   <b_list> → <b_list> , <bind>   S

20   <b_list> → <bind>   S

1

21   <bind> → st_id : (<rel_exp>)   S

22   <bind> → st_id : (<rel_exp>)   S

23   < st_binding> → Є   S

24   <cls_m_list> → <cls_m_list> <cls_method>

25   <cls_m_list> → Є

26   <cls_method> → <pub-prv> <r_type> <m_id> (<arg_list>) { <m_body> }

27   <pub-prv> → public

28   <pub-prv>→ private

29   <r_type> → <dt>

30   <m_id> → m_id

31   <arg_list> → Є

32   <arg_list> → <args>

33   <args>→ <args>, <arg>

34   <args>→<arg>

35   <arg> →<dt> id

36   <m_body> → <var_dec_list> <stm_list>

37   <stm_list> → <stm_list> <stm>

38   <stm_list> → Є

39   <stm> → <lhs> = <exp>;

40   <stm> → read (id) ;

41   <stm> → print <exp>;

42   <stm> → o_id.m_id(<param_list>);

43   <vd_new> → = new ID ()

44   <stm> → if (<rel_exp>) {<stm_list> } <else>

45   <else>→ else {<stm_list>}

46   <else> → Є

47   <stm> →while(<rel_exp>){<stm_list> }

48   <stm>→for(<init><mid>;<last>) { <stm_list>}

49   <rel_exp>→<rel_exp> <rel_op> <rel_term>

50   <rel_term> →<rel_bool_fac>

51   <rel_exp> →<rel_term>

52   <rel_term> → <rel_fac><comp_op><rel_fac>

53   <rel_bool_fac> → bool_id

54   <rel_fac> → id

55   <rel_fac> → num

56   < rel_fac > → o_id.p_id

57   <rel_op> → AND

58   <rel_op> → OR

59   <comp_op> → ==

60   <comp_op> → !=

61   <comp_op> → >

62   <comp_op> → <

63   <comp_op> → <=

64   <comp_op> → >=

65 <exp> → <term>

66 <exp> → <exp> <add_op> <term>

67 <term> → <factor>

68 <term> → <term> <mult_op> <factor>

69 <factor> → (<exp>)

70 <factor> → id

71 <factor> → o_id.p_id

72 <factor> → num

73 <add_op> → +

74 <add_op> → -

75 <mult_op> → *

76 <mult_op> → /

77 <c_s_m_list> → Є

78 <c_s_m_list> → <c_s_m_list> <c_s_method>   S

79 <c_s_method> → <s_list> { <s_v_d_lst><s_m_list> }   S

80 <s_m_list> → <s_m_list> <s_method>   S

81 <s_m_list> → <s_method>   S

82 <s_method> → <r_type><m_id>(<arg_list>)<s_end_list>{<s_m_body>} S

83 <s_end_list> → [<s_e_list>]   S

84 <s_list> → <s_list> , <s_id>   S

85 <s_list> → <s_id>   S

86 <s_e_list> → <s_e_list> | st_id   S

87	<s_e_list> → st_id	S
88	<s_m_body> →<var_dec_list> <s_stm_list>	
89	<s_stm_list> →<s_stm_list> <st_stm>	
90	<s_stm_list> → Є	
91	<st_stm>→this.trans(st_id);	S
92	<st_stm>→<lhs> = <exp>;	
93	<st_stm>→ read (id) ;	
94	<st_stm>→ print <exp>;	
95	<st_stm> →o_id.m_id(<param_list>);	
96	<st_stm> →o_id.m_id(<param_list>);	
97	<st_stm>→ while(<rel_exp>){<s_stm_list> }	
98	<st_stm>→ for(<init><mid>;<last>) { <s_stm_list> }	
99	< st_stm> → if (<rel_exp>) {<s_stm_list>}<s_else>	
100	<s_else> → else {<s_stm_list>}	
101	<s_else> → Є	
102	<init>→ id = num;	
103	<mid>→<rel_exp>	
104	<last>→id <in_dec>	
105	<in_dec>→ ++	
106	<in_dec>→ --	
107	<lhs>→id	
108	<lhs>→ o_id.p_id	

5

109	<param_list>→<params>
110	<params>→<params>,<factor>
111	<params>→<factor>
112	<param_list>→ ϵ
113	<new>→ ϵ
114	<cl_id>→ cl_id
115	<s_id> → [st_id]	S
116	<s_list> → [st_id]:[st_id]	S
117	<vd_rest> → <vd_simple>	S
118	<vd_rest> →<vd_new>	S
119	<vd_simple> → <vd_simple> , id	S
120	<vd_simple> → ϵ	S
121	<vd_new> → ϵ	S
122	<s_v_d_lst> → <s_v_d_lst><s_v_d> ;	S
123	<s_v_d_lst> → ϵ	S
124	<s_v_d>→<s_vd_start><vd_rest>	S
125	< s_vd_start> → <dt> id	S