版本: flex 2.6.4, GNU Bison 3.8.2
作業平台: Ubuntu 22.04.5 (VirtualBox)
執行方式: Type following on terminal : ./b113040002< <filename>

規格書要求:

我的文法結構:

1. Program
    a. class declaration
        i. variable declaration
        ii. method declaration
            1. variable declaration
            2. class declaration
            3. statement
                a. if-else
                b. for loop
                c. while loop
                d. print
                e. return
                f. expression
                    i. prefix op expr
                    ii. postfix op expr
                    iii. expr operator expr
                    iv.
        iii. class declaration
            1. variable declaration
            2. class declaration
            3. method declaration

此結構為參考真實JAVA的程式結構做成
此次並沒有回傳comment**和**NEWLINE**的**token**，如同真的**java**一樣忽視他們

個別文法:

1. class declaration

```
class_list:
    class_list class_decl{/*printf("sd\n");*/}
  | class_decl{/*printf("sd\n");*/}
;

class_decl:
    CLASS ID
    {
        //printf("ds\n");
        if ( add_symbol($2) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $2);
            free($2);
        }
    }
    LBRACE
   {enter_scope();}
    class_body_opt
   {exit_scope();}
    RBRACE
;

class_body_opt:
    /* empty */
  | nonempty_class_body
;
```

```
class_body_opt:
    /* empty */
  | nonempty_class_body
;

nonempty_class_body:
    nonempty_class_body member_decl /*{printf("bodyline\n");}*/
  | member_decl/*{printf("bodyline\n");}*/
;

member_decl:
    var_decl
  | STATIC var_decl
  | FINAL var_decl
  | CONST var_decl
  | ACCESS_MODIFIER var_decl
  | method_decl /*{printf("to member_decl\n");}*/
  | FINAL method_decl
  | CONST method_decl
  | ACCESS_MODIFIER method_decl
  | STATIC method_decl
  | class_decl       /* 支援巢狀類別 */
;
```

擅自用讓nonterminal變空容易使觸發RS衝突，因此在允許多行的迴圈中使其必定會輸出至少一行。menber declaration中將所有前綴token列出也是為了免衝突

***遇到困難1：non-terminal變空容易發生衝突，因此在處理重複出現時一條文法最好是限定至少出現一次，在其上層再設定此文法能否出現。

2. variable declaration

```
var_decl:
    ID var_list SEMICOLON
  | ID var_list {
      unsigned D = charCount - yyleng;
      printf("ERROR*****Line %d, char: %d: Expect ';' at end of declaration.\n", lineCount, D);
      yyerrok;
    }error

  | ID LBRACKET RBRACKET var_list SEMICOLON
  | TYPE var_list SEMICOLON
  | TYPE LBRACKET RBRACKET var_list SEMICOLON
  | TYPE var_list error var SEMICOLON
    {
      printf("ERROR*****Line %d, char: %d: Expect ',' before '%s'.\n", lineCount, charCount, $4);
      free($4);
      yyerrok;
    }
;

var_list:
    var_list COMMA var
  | var
;
```

```
var:
    ID
    {
        if ( add_symbol($1) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $1);
            free($1);
        }
      $$ = $1;
    }
  | ID ASSIGN expression
    {
          if ( add_symbol($1) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $1);
            free($1);
        }
      $$ = $1;
    }
  | ID ASSIGN error
    {
          if ( add_symbol($1) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $1);
            free($1);
        }
      $$ = $1;
       printf("ERROR*****Line %d, char: %d: Illigel variable assigment.\n", lineCount, charCount);
       yyerrok;
    }
;
```

變數在一次宣告相同類型的變數時採取了同樣至少出現一次的處理以避免衝突。然後為了處理測使檔的錯誤，特別分出了沒有分號結尾的情況以及變數間缺失逗號的情況。除此之外還有等號後什麼都沒有的情況。

***遇到困難2 :Modifier 如果出現在這層會需要設立為空的情況，這樣的狀況會使衝突可能出現，因此我移到上層。

如果將動作區插進token之間會影響傳值($$)的對象，因此動作必須移到整個文法的最後。

3. method declaration

```
method_decl:
     TYPE ID LP param_list_opt RP block
    {
        if ( add_symbol($2) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $2);
        }
        free($2);
    }
  | ID LP param_list_opt RP block
    {
        if ( add_symbol($1) < 0 ) {
            printf("ERROR*****Line %d, char: %d: '%s' is a duplicate identifier.\n", lineCount, charCount, $1);
        }
        free($1);
    }
  | MAIN LP param_list_opt RP block
  | TYPE MAIN LP param_list_opt RP block
;

param_list_opt:
    /* empty */
  | param_list
;

param_list:
    param_list COMMA TYPE ID
    {
        add_symbol($4);
        free($4);
    }
  | TYPE ID
    {
        add_symbol($2);
        free($2);
    }
;
```

參數可能會出現無限次，因此一樣是分成 ”會出現”vs”不會出現” 層，”出現1次” vs “出現多次” 一層。而且main不是 identifier 因此自己獨立分出來。

4. Block

```
block:
     LBRACE
   {enter_scope();}
    block_items_opt
   {exit_scope();}
    RBRACE
;

block_items_opt:
    /* empty */
  | nonempty_block_items
;

nonempty_block_items:
    nonempty_block_items block_item
  | block_item
;

block_item:
    var_decl
  | statement
  | class_decl
;
```

Block 指的是單一scope中的一切，屬於 statement中特殊的存在，不但可以單獨出現，也可以出現在 if-else、loop、method等處重複使用，因此將其獨立出來。block的結構和class十分相似，同樣有因應本地全域的問題，也有對應重複出現的迴圈的設計。

5. statement

```
statement:
    IF LP expression RP statement       %prec LOWER_THAN_ELSE
  | IF LP expression RP statement ELSE statement
  | ELSE statement
    {
      printf("ERROR*****Line %d, char: %d: Else without if statement.\n", lineCount, charCount);
    }
  | WHILE LP expression RP
    {is_loop = 1;}
    statement
    {is_loop = 0;}
  | WHILE LP error RP {
      {
        printf("ERROR*****Line %d, char: %d: Invalid boolean expression.\n", lineCount, charCount);
        yyerrok;
      }
    }statement

  | FOR LP for_init_opt SEMICOLON expression SEMICOLON for_update_opt RP
    {is_loop = 1;}
    statement
    {is_loop = 0;}
  | FOR LP for_init_opt SEMICOLON SEMICOLON for_update_opt RP
    {printf("ERROR*****Line %d, char: %d: For loop will not stop.\n", lineCount, charCount);
     is_loop = 1;}
    statement
    {is_loop = 0;}
  | RETURN expression SEMICOLON
  | PRINT LP expression RP SEMICOLON
  | expr_stmt
  | block
  | BREAK_CONTINUE SEMICOLON
    {
        if(is_loop != 1) printf("Break / Continue is not in loop scope.\n");
    }
;
```

```
for_init_opt:
      /* empty */
    | TYPE ID ASSIGN expression
      {
          add_symbol($2);
          free($2);
      }
    | expression
;
```

具有本次測試檔出現的所有文法，主要依照規則書上的舉例建構，為了對應衝突的可能，if-else分成三種類別。For 迴圈會偵測有沒有終止條件，並在沒有時輸出提示。同時while也增加了處理 error boolean expression的分支。Continue和break也增加偵測自己是否在loop中的設計，如果不是，會輸出錯誤提示。

6. expression

```
expr_stmt:
     expression SEMICOLON
   | expression error
     {
        printf("ERROR*****Line %d, char: %d: Expect ';' at end of line.\n", lineCount, charCount);
        yyerrok;
     }
;
```

```
expression:
     expression ADD  expression
   | expression MINUS expression
   | expression MUL   expression
   | expression DIV   expression
   | expression MOD   expression
   | expression EQ    expression
   | expression NE    expression
   | expression LT    expression
   | expression LE    expression
   | expression GT    expression
   | expression GE    expression
   | expression ASSIGN expression
   | expression ASSIGN error
     {
        printf("ERROR*****Line %d, char: %d: Illigel variable assigment.\n", lineCount, charCount);
        yyerrok;
     }
   | expression INC
   | INC expression
   | expression DEC
   | DEC expression
   | primary
;
```

```
primary:
     ID %prec LOWER_THAN_LBRAKET
     {
         if ( find_symbol($1) < 0 ) {
             printf("ERROR*****Line %d, char: %d: '%s' is undeclared.\n", lineCount, charCount, $1);
             free($1);
         }
     }
   | NUMBER
   | STRING
   | ID
     {
         if ( find_symbol($1) < 0 ) {
             printf("ERROR*****Line %d, char: %d: '%s' is undeclared.\n", lineCount, charCount, $1);
             free($1);
         }
     }LP arg_list_opt RP
   | ID LBRACKET {
         if ( find_symbol($1) < 0 ) {
             printf("ERROR*****Line %d, char: %d: '%s' is undeclared.\n", lineCount, charCount, $1);
             free($1);
         }
     } expression RBRACKET

   | NEW ID
     {
         if ( find_symbol($2) < 0 ) {
             printf("ERROR*****Line %d, char: %d: '%s' is undeclared.\n", lineCount, charCount, $2);
             free($2);
         }
     }
   | NEW ID
     {
         if ( find_symbol($2) < 0 ) {
             printf("ERROR*****Line %d, char: %d: '%s' is undeclared.\n", lineCount, charCount, $2);
             free($2);
         }
     }LP arg_list_opt RP
   | NEW TYPE LBRACKET expression RBRACKET
   | LP expression RP
;
```

expression的第一層是對應直接當作statement的裝況，因此會特別留意結尾的分號。且具有賦值的文法也追加了防止等號後沒東西的狀況。第二層則是可能出現的情況，對於會調用變數的文法都有加設預防變數未宣告的提示。

輸出結果:

Test1

```
vboxuser@Ubuntu22:~$ ./yacc<test1.java
line 1: /* Test file: Perfect test file
line 2:  * Compute sum = 1 + 2 + ... + n
line 3:  */
line 4: class sigma {
line 5: // "final" should have const_exprline 5:
line 6:   final int n = 10;
line 7:   int sum, index;
line 8:
line 9:   main()
line 10:   {
line 11:     index = 0;
line 12:     sum = 0;
line 13:     while (index <= n)
line 14:     {
line 15:       sum = sum + index;
line 16:       index = index + 1;
line 17:     }
line 18:     print(sum);
line 19:   }
line 20: }
```

Test2

```
line 1: /*Test file: Duplicate declare variable in the same scope*/
line 2: class Point
line 3: {
line 4:     static int counter ;
line 5:     int x, y ;
line 6: /*Duplicate declare x*/
ERROR*****Line 7, char: 30: 'x' is a duplicate identifier.
line 7:     int x ;
line 8:     void clear()
line 9:     {
line 10:         x = 0 ;
line 11:         y = 0 ;
line 12:     }
line 13: }
```

Test3

```
vboxuser@Ubuntu22:~$ ./yacc<test3.java
line 1: /*Test file of Syntax errer: Out of symbol.  But it can go through*/
line 2: class Point {
line 3:     int z;
Syntax error at line 4
ERROR*****Line 4, char: 39: Expect ',' before 'y'.
line 4:     int x y ;
line 5: /*Need  ',' before y*/
line 6:     float w;
line 7: }
line 8: class Test {
line 9:     int d;
line 10:     Point p = new Point()
line 11: /*Need ';' at EOL*/
ERROR*****Line 12, char: 9: Expect ';' at end of declaration.
Syntax error at line 12
line 12:     int w,q;
line 13: }
```

Test4

```
vboxuser@Ubuntu22:~$ ./yacc<test4.java
line 1: /*Test file: Duplicate declaration in different scope and same scope*/
line 2: class Point
line 3: {
line 4:         int x, y ;
line 5:         int p;
line 6:         boolean test()
line 7:         {
line 8: /*Another x, but in different scopes*/
line 9:                 int x;int ds;
line 10:                int z;
line 11:
line 12: /*Another x in the same scope*/
ERROR*****Line 13, char: 23: 'x' is a duplicate identifier.
line 13:                char x;
line 14:                {
line 15:                        boolean w;
line 16:                }
line 17: /*Another w in the different scope*/
line 18:                int w;
line 19:        }
line 20: }
line 21: class Test
line 22: {
line 23: /*Another p, but in different scopes*/
line 24:        Point p = new Point();
line 25: }
```

Test5

```
vboxuser@Ubuntu22:~$ ./yacc<test5.java
line 1: class test5{
line 2:     int add(int a1, int a2){
line 3:         return (a1 + a2);
line 4:     }
line 5:     void main() {
line 6:         int x, y, z;
line 7:         for(int i=0;i<2;i++){
line 8:             if(i==0){
line 9: //--------------ELSE WITHOUT IFline 9:
line 10:                else
ERROR*****Line 11, char: 29: Else without if statement.
line 11:                    i = 1;
line 12:            }
line 13:            for(x = 0; x<5;x++){
line 14:                y++;
line 15: //--------------FUNCTION CALLline 15:
line 16:                x = add(x,y);
line 17:                x = z(x,y);
line 18:            }
line 19:        }
line 20:        print("x:"+x+"y:"+y);
line 21:        z = ( x + y ) * 5 / 2 -- -y;
line 22:    }
line 23: }
line 24:
line 25: /* this is a comment // line// with some /* /*and
line 26: // delimiters */
```

\

Test6

```
line 1: class test6{
line 2:     void sum(){
line 3: //------NEVER USEDline 3:
ERROR*****Line 4, char: 22: 'x' is undeclared.
ERROR*****Line 4, char: 26: 'y' is undeclared.
ERROR*****Line 4, char: 30: 'z' is undeclared.
line 4:         int sumxyz = x + y + z ;
line 5:     }
line 6:     void main() {
line 7: //------ARRAYline 7:
line 8:         int [] i= new int [1];
line 9:         for(i[0] = 0; i[0]<5; i[0]++)
line 10:             i[0]++;
line 11:
line 12: //------NEW CLASSline 12:
ERROR*****Line 13, char: 31: 'Point' is undeclared.
ERROR*****Line 13, char: 9: 'Point' is undeclared.
line 13:         Point lowerLeft = new Point() ;
line 14:
line 15: //------ERROR CONDITIONline 15:
Syntax error at line 16
ERROR*****Line 16, char: 22: Invalid boolean expression.
line 16:         while(**/a++){
line 17:             print("error!!");
line 18:         }
line 19: //------CLASS DECLAREline 19:
line 20:         class Point {
line 21:             int x, y, z;
line 22:         }
line 23:     }
line 24:
line 25: }
```