Simple Java — Parser

學號：B113040052

姓名：陳育霖

Lex 版本：

使用版本：flex 2.6.4

Yacc 版本：

使用版本：Bison 3.8.2

作業平台

作業系統：Ubuntu 22.04.4 LTS

編譯器：gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

編譯與執行方式

1. **一鍵編譯**

   make

2. **測試執行**

   make test

3. **清除檔案**

   make clean

4. **執行 yacc.elf 測試**

   ./yacc.elf < test/test1.java

規格書處理方式

1. 詞彙分析（Lex）

- 保留字與型別識別：
  使用正規表示式定義 class, int, boolean, new, if, else, while, return 等保留字，並回傳對應的 token。
- **ID 與常數：**
  - ID：支援以字母開頭、後接數字或底線的命名規則。
  - 常數：定義整數與 true、false 字面值的辨識。
- **符號追蹤與錯誤行定位：**
  - 自訂 SET_TOKEN_INFO() 巨集儲存 token_col, last_token_col 等資訊，用於精確報錯。
  - 每掃描一個 token，自動計數 charCount, lineCount，更新行緩衝。
- **錯誤偵測（非法字元與運算符）：**
  - 如遇 **/、@ 等非法符號，會設定 operator_error_reported = 1，並呼叫 yyerror()。
  - 特殊處理空白與註解跳過，同時保持位置資訊正確。

2. 語法分析（Yacc）

- **起始符號與程式結構**：
  - 以 %start classes 與 lines 為根規則，支援多行類別與語句解析。
  - 使用 classDecl, functionDecl, varDecl 等語法規則建構程式結構。
- **符號表與作用域管理**：
  - 實作 push() / pop() 管理巢狀作用域，每遇新類別或函式自動進入新層級。
  - 透過 insert_entry() 與 lookup_local() 控制重複定義。
- **錯誤處理機制**：
  - 在 lines 規則中加入 | error '\n' { flushPendingError(); }，遇錯誤自動跳至下一行。
  - 使用 skip_next_decl_typecheck 與 error_reported 標記跳過錯誤宣告的語意檢查。
- **分號與結構缺失偵測**：
  - 對於 Point p = new Point() 無結尾分號的情況，會在下個 token 或宣告時報錯。
  - 配合 flushPendingError() 延遲輸出以對齊原始程式行。

- **複合語法處理**：
    - 陣列：如 int[] a = new int[3]; 透過 type '[' ']' 與 newExp 規則處理。
    - 運算式：支援布林運算、比較運算、加減乘除，透過優先順序與結合律處理歧義。
    - 建構式與物件建立：如 new Point() 經由 newExp 與 call 語法匹配處理。

這個作業所遇到的問題

問題 1

當輸入程式缺少分號或含有語法錯誤時，會立即中止並停止後續解析，導致僅能回報第一個錯誤，無法分析整份程式碼。

解法：

在 lines 規則中加入錯誤同步處理：

lines: lines line

| error '\n' { flushPendingError(); yyerrok; }

在發生語法錯誤時跳過該行，並嘗試繼續處理後續程式碼，同時搭配 flushPendingError() 延後輸出錯誤訊息，使其行數正確對齊。

問題 2

當同名變數或函式在不同區塊中宣告（例如在類別內與函式內），會被誤判為重複定義。

解法：

導入多層次符號表的機制，使用 parse_depth 配合 push()/pop() 控制進出作用域層級，並分別檢查 local/global：

lookup_local() // 檢查目前區塊是否重複

insert_entry() // 僅當 lookup_local 無結果時才插入

問題 3

由於錯誤可能出現在前一行的結尾，但錯誤訊息卻報在下一行。

解法：

實作 last_token_line 與 last_token_col，在每個 token 被讀取時紀錄前一個 token 的位置。

測試檔執行出來的結果

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_expr
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.java

```
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*/
Line 7, char: 12, duplicate identifier 'x'
line 7: int x ;
line 8: void clear ( )
line 9: {
line 10: x = 0 ;
line 11: y = 0 ;
line 12: }
line 13: }
```

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 ;
Line 4, char: 12, syntax error: expected semicolon
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*/
Line 10, char: 29, statement without semicolon
line 12: int w , q ;
line 13: }
```

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 ;
line 10: /*Another x in the same scope*/
Line 11, char: 16, duplicate identifier 'x'
line 11: char x ;
line 12: {
line 13: boolean w ;
line 14: }
line 15: /*Another w in the same scope*/
line 16: int w ;
line 17: }
line 18: }
line 19: class Test
line 20: {
line 21: /*Another p, but in different scopes*/
line 22: Point p = new Point ( ) ;
line 23: }
```

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 IF
line 10: else
Line 10, char: 17, 'else' without matching 'if'
line 11: i = 1 ;
line 12: }
line 13: for ( x = 0 ; x < 5 ; x ++ ) {
line 14: y ++ ;
line 15: //--------------FUNCTION CALL
line 16: x = add ( x , y ) ;
Line 17, char: 33, undefined function 'z'
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.java

```
line 1: class test6 {
line 2: void sum ( ) {
line 3: //------NEVER USED
line 4: int sumxyz = x + y + z ;
line 5: }
line 6: void main ( ) {
line 7: //------ARRAY
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 CLASS
Line 13, char: 58, undefined function 'Point'
Line 13, char: 20, a syntax error at "lowerLeft"
line 13: Point lowerLeft = new Point ( ) ;
line 14:
line 15: //------ERROR CONDITION
Line 16, char: 14, invalid token '**/'
Line 16, char: 9, a syntax error at "while"
line 16: while ( a ++ ) {
Line 17, char: 13, a syntax error at "print"
line 17: print ( "error!!" ) ;
line 18: }
line 19: //------CLASS DECLARE
line 20: class Point {
line 21: int x , y , z ;
line 22: }
line 23: }
line 24:
Line 25, char: 1, unmatched '}'
line 25: }
```