Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών
Εργαστηριακή Άσκηση 2024

Γκιριτζιώνη Παναγιώτα 1070953 7ο
st1070953@ceid.upatras.gr

Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

1) BNF1:

<program> ::= <class_declaration> <code>

<code> ::= <class_declaration> <code>
| <method_declaration> <code>
| <variable_declaration> <code>
| <print_command> <code>
| ε

<class_declaration> ::= <begin_class_declaration> "{" <class_body> "}"

<begin_class_declaration> ::= <access_modifier> "CLASS" <class_name>

<class_body> ::= <variable_declaration> <class_body>
| <method_declaration> <class_body>
| <class_declaration> <class_body>
| <object_creation> <class_body>
| ε

<class_member> ::= <class_name> "." <identifier>
| <class_name> "." <identifier> "(" <method_call_parameter_list> ")"

<variable_declaration> ::= <access_modifier> <data_type> <identifier> ";"

<opt_variable_declarations> ::= <opt_variable_declarations> <variable_declaration>

<method_declaration> ::= <access_modifier> <data_type> <identifier> "("
<method_parameter_list> ")" "{" <method_body> "}"
| <access_modifier> "VOID" <identifier> "(" <method_parameter_list> ")" "{"
<method_body> "}"

<method_parameter_list> ::= <method_parameter_list> "," <method_parameter>
| <method_parameter>
| ε

<method_call_parameter_list> ::= <method_call_parameter_list> "," <method_call_parameter>
| <method_call_parameter>
| ε

<method_parameter> ::= <data_type> <identifier>

<method_call_parameter> ::= <identifier>

<method_body> ::= <opt_variable_declarations> <commands> <opt_return>

<opt_variable_declarations> ::= <opt_variable_declarations> <variable_declaration>
| ε

<commands> ::= <command> <commands>
| ε

<opt_return> ::= "return" <expression> ";"
| "return" ";"
| ε

<access_modifier> ::= "public"
| "private"
| ε

<commands> ::= <command> <commands>
| ε

<commands_with_break> ::= <commands>
| <commands> "BREAK" ";"

<command> ::= <print_command>
| <assign_command>
| <object_creation>
| <if_statement>
| <switch_statement>
| <do_loop>
| <for_loop>
| <comment>

<object_creation> ::= <class_name> <identifier> "=" <object_instation> ";"

<print_command> ::= "PRINT.OUT" "(" "STRING_LITERAL" ")" ";"
| "PRINT.OUT" "(" "STRING_LITERAL" "," <method_call_parameter_list> ")" ";"

<assign_command> ::= <identifier> "=" <expression> ";"
| <class_member> "=" <expression> ";"

<assign_command_no_semicolon> ::= <identifier> "=" <expression>
| <class_member> "=" <expression>

<class_member> ::= <class_name> "." <identifier>
| <class_name> "." <identifier> "(" <method_call_parameter_list> ")"

<object_instation> ::= "NEW" <class_name> "(" ")"

<method_call_parameter_list> ::= <method_call_parameter_list> "," <method_call_parameter>
| <method_call_parameter>
| ε

<method_call_parameter> ::= <identifier>

<if_statement> ::= "IF" "(" <bool_expression> ")" "{" <commands_with_break> "}" <else_if_list>
<else_statement>

<else_if_statement> ::= "ELSE IF" "(" <bool_expression> ")" "{" <commands_with_break> "}"

<else_if_list> ::= <else_if_list> <else_if_statement>
| ε

<else_statement> ::= "ELSE" "{" <commands_with_break> "}"
| ε

<switch_statement> ::= "SWITCH" "(" <expression> ")" "{" <case_statement_list> <opt_default> "}"

<case_statement> ::= "CASE" <expression> ":" <commands>

<case_statement_list> ::= <case_statement_list> <case_statement>
| <case_statement>

<opt_default> ::= "DEFAULT" ":" <commands>
| ε

<do_loop> ::= "DO" "{" <commands_with_break> "}" "WHILE" "(" <bool_expression> ")"

<for_loop> ::= "FOR" "(" <variable_declaration_with_assign> ";" <comparison_expression> ";"
<assign_command_no_semicolon> ")" "{" <commands_with_break> "}"

<expression> ::= <numerical_expression>
| <bool_expression>
| "CHAR_LITERAL"
| "STRING_LITERAL"
| <object_instation>

```
            | <identifier> "(" <method_call_parameter_list> ")"

<object_instation> ::= "NEW" <class_name> "(" ")"

<numerical_expression> ::= "INT_LITERAL"
                    | <double_literal>
                    | <identifier>
                    | <class_member>
                    | "(" <numerical_expression> ")"
                    | "-" <numerical_expression>
                    | <numerical_expression> "+" <numerical_expression>
                    | <numerical_expression> "-" <numerical_expression>
                    | <numerical_expression> "*" <numerical_expression>
                    | <numerical_expression> "/" <numerical_expression>

<bool_expression> ::= <bool_literal>
                | <comparison_expression>
                | "(" <bool_expression> ")"
                | <bool_expression> "AND" <bool_expression>
                | <bool_expression> "OR" <bool_expression>

<comparison_expression> ::= "(" <comparison_expression> ")"
                    | <numerical_expression> ">" <numerical_expression>
                    | <numerical_expression> "<" <numerical_expression>
                    | <numerical_expression> "EQ" <numerical_expression>
                    | <numerical_expression> "NEQ" <numerical_expression>

<bool_literal> ::= "TRUE"
                | "FALSE"

<double_literal> ::= "DOUBLE_LITERAL" "DOUBLE_SP"


<opt_return> ::= "RETURN" <expression> ";"
            | "RETURN" ";"
            | ε

<identifier> ::= <class_name>
            | <generic_identifier>

<data_type> ::= "INT"
           | "DOUBLE"
           | "BOOLEAN"
           | "CHAR"
           | "STRING"
```

```
<access_modifier> ::= "PUBLIC"
                    | "PRIVATE"
                    | ε
<class_name> ::= <capital_letter> <identifier_tail>

<generic_identifier> ::= <identifier_start> <identifier_tail>
<identifier_start> ::= <letter> | "_"
<identifier_tail> ::= <identifier_char> <identifier_tail>
                    | ε
<identifier_char> ::= <letter>
                    | <digit>
                    | "_"

<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
<capital_letter> ::= "A" .. "Z"
<digit> ::= "0" | "1" | ... | "9"


<comment> ::= <single_line_comment> | <multi_line_comment>
<single_line_comment> ::= "//" <any_character_sequence> "\n"
<multi_line_comment> ::= "/*" <any_character_sequence> "*/"
```

FLEX:

```
%option yylineno

%{
#include <cstdlib>
#include <iostream>

#include "../parser_helper.h"
#include <string>
#include "bison_parser.tab.hpp"

%}

%%

[ \t\r\n]+          /* Αγνόησε τα κενά */;

"//".*              { /* Αγνόησε σχόλιο μίας γραμμής */ }
"/*"([^*]|\*[^/])*"*/"  { /* Αγνόησε σχόλιο πολλαπλών γραμμών */ }

"INT"           { return INT; }
"CHAR"          { return CHAR; }
```

```
"DOUBLE"          { return DOUBLE; }
"BOOLEAN"         { return BOOLEAN; }
"STRING"          { return STRING; }
"CLASS"           { return CLASS; }
"NEW"             { return NEW; }
"RETURN"          { return RETURN; }
"VOID"            { return VOID; }
"IF"              { return IF; }
"ELSE"            { return ELSE; }
"WHILE"           { return WHILE; }
"DO"              { return DO; }
"FOR"             { return FOR; }
"SWITCH"          { return SWITCH; }
"CASE"            { return CASE; }
"DEFAULT"         { return DEFAULT; }
"BREAK"           { return BREAK; }
"TRUE"            { return TRUE; }
"FALSE"           { return FALSE; }
"PUBLIC"          { return PUBLIC; }
"PRIVATE"         { return PRIVATE; }

"PRINT"           { return PRINT; }
"OUT"             { return OUT; }
"d"               { return DOUBLE_SP; }

"="               { return '='; }
"+"               { return '+'; }
"-"               { return '-'; }
"*"               { return '*'; }
"/"               { return '/'; }
">"               { return '>'; }
"<"               { return '<'; }
"=="              { return EQ; }
"!="              { return NEQ; }
"&&"              { return AND; }
"||"              { return OR; }
"."               { return '.'; }
"("               { return '('; }
")"               { return ')'; }
"{"               { return '{'; }
"}"               { return '}'; }
";"               { return ';'; }
","               { return ','; }
":"               { return ':'; }    // For switch/case

[0-9]+"."[0-9]*     { yylval.double_val = std::atof(yytext); return DOUBLE_LITERAL; }
```

```
[0-9]+            { yylval.int_val = std::atoi(yytext); return INT_LITERAL; }

'[^']'            { yylval.char_val = yytext[1]; return CHAR_LITERAL; }      // Χαρακτήρες
'\\[a-zA-Z]'      { yylval.char_val = yytext[2]; return CHAR_LITERAL; }      // Ειδικοί Χαρακτήρες
\"([^\\"]|\\.)*\"   { yylval.str_val = strdup(yytext); return STRING_LITERAL; } // Strings

[A-Z][a-zA-Z0-9_]*      { yylval.str_val = strdup(yytext); return CLASS_NAME; }     // Ονόματα κλάσεων (ξεκινούν με κεφαλαίο)
[a-zA-Z_][a-zA-Z0-9_]*  { yylval.str_val = strdup(yytext);  return IDENTIFIER; }    // Γενικά αναγνωριστικά


<<EOF>>           { return 0; }      // Τέλος αρχείου
.                 { std::cerr << "Unexpected character: " << yytext << std::endl; return 0; }

%%

int yywrap() {
   return 1;  // Indicate end of input to the parser
}
```

BISON:

```
%require "3.5.1"

%define parse.error verbose

%{
   #include <iostream>
   #include <string>
   #include "../parser_helper.h"

   extern int yylex();
   extern int yyerror(const char *s);
   extern int yylineno;

   auto parser_helper = new ParserHelper();

   DataType temp_data_type;
   bool temp_access_modifier;
%}

%union {
   int int_val;
```

```
   double double_val;
   bool bool_val;
   char char_val;
   char* str_val;
}

/*********************************
 ** Tokens
 *********************************/
%token <int_val>       INT_LITERAL;
%token <double_val>    DOUBLE_LITERAL;
%token <char_val>      CHAR_LITERAL;
%token <str_val>       IDENTIFIER CLASS_NAME STRING_LITERAL;

%token  EQ NEQ AND OR;

%token  PRINT OUT DOUBLE_SP

%token  INT CHAR DOUBLE BOOLEAN
       STRING CLASS NEW RETURN
       VOID IF ELSE WHILE DO FOR
       SWITCH CASE DEFAULT BREAK
       TRUE FALSE PUBLIC PRIVATE;


/*********************************
 ** Non-Terminal types
 *********************************/

%nterm <double_val>    double_literal;
%nterm <bool_val>      bool_literal access_modifier;
%nterm <str_val>       identifier begin_class_declaration;

// Ο πρώτος κανόνας ορίζει τι πρεπει να δεχεται ο parser
%start program

%left '+' '-'
%left '*' '/'
%left '>' '<' EQ NEQ
%left AND OR
%left '.'

%%

// Το προγραμμα αποτελείται απο μια η περισσότερες classes και επιπλέον κώδικα
program:
```

```
        class_declaration code
    ;

code:
        class_declaration code
    |   method_declaration code
    |   variable_declaration code
    |   print_command code
    |   %empty


// ------------------------------ Declarations ------------------------------

class_declaration:
        begin_class_declaration '{' class_body '}'
          {
            if ( !parser_helper->in_class.empty() ) {
              parser_helper->in_class.pop_back();
            }
          }
    ;

begin_class_declaration:
        access_modifier CLASS CLASS_NAME
          {
            parser_helper->addClass($3, $1);
            parser_helper->in_class.push_back($3);
          }
    ;

class_body:
        variable_declaration class_body
    |   method_declaration class_body
    |   class_declaration class_body
    |   object_creation class_body
    |   %empty
    ;

class_member:
        CLASS_NAME'.'identifier
          {
            const auto opt_error = parser_helper->getVariable($1, $3);
            if(opt_error){
              yyerror(opt_error.value().c_str());
              YYERROR;
            }
```

```
        }
  |   CLASS_NAME'.'identifier'(' method_call_parameter_list ')'
        {
           const auto opt_error = parser_helper->getMethod($1, $3);
           if(opt_error){
              yyerror(opt_error.value().c_str());
              YYERROR;
           }
        }
  ;



variable_declaration:
    access_modifier data_type identifier ';'
     {
        parser_helper->addVariable($3, temp_data_type, $1);
     }
  ;



opt_variable_declarations:
       opt_variable_declarations variable_declaration
  |   %empty
  ;

method_declaration:
       access_modifier data_type identifier '(' method_parameter_list ')' '{' method_body '}'
         {
            parser_helper->addMethod($3, temp_data_type ,$1);
         }
  |   access_modifier VOID identifier '(' method_parameter_list ')' '{' method_body '}'
         {
            parser_helper->addMethod($3, DataType::VOID_TYPE,$1);
         }
  ;

method_parameter_list:
       method_parameter_list ',' method_parameter
  |   method_parameter
  |   %empty
  ;

method_call_parameter_list:
```

```
        method_call_parameter_list ',' method_call_parameter
    |   method_call_parameter
    |   %empty
    ;

method_parameter:
        data_type identifier
    ;

method_call_parameter:
        identifier
    ;

method_body:
        opt_variable_declarations commands opt_return
    ;


// ------------------------------ Commands ------------------------------

commands:
        command commands
    |   %empty
    ;

commands_with_break:
        commands
    |   commands BREAK ';'
    ;

command:
        print_command
    |   assign_command
    |   object_creation
    |   if_statement
    |   switch_statement
    |   do_loop
    |   for_loop
    ;

object_creation:
        CLASS_NAME identifier '=' object_instation ';'
    ;

print_command:
        PRINT'.'OUT '(' STRING_LITERAL ')' ';'
```

```
    |   PRINT'.'OUT '(' STRING_LITERAL ',' method_call_parameter_list')' ';'
    ;

assign_command:
        identifier '=' expression ';'
        {
            const auto opt_error = parser_helper->getVariable(parser_helper->in_class.back(), $1);
            if(opt_error){
                yyerror(opt_error.value().c_str());
                YYERROR;
            }
        }
    |   class_member '=' expression ';'
    ;

assign_command_no_semicolon:
        identifier '=' expression
    |   class_member '=' expression
    ;

// ------------------------------ If & Switch ------------------------------

if_statement:
        IF '(' bool_expression ')' '{' commands_with_break '}' else_if_list else_statement
    ;

else_if_statement:
        ELSE IF '(' bool_expression ')' '{' commands_with_break '}'
    ;

else_if_list:
        else_if_list else_if_statement
    |   %empty
    ;

else_statement:
        ELSE '{' commands_with_break '}'
    |   %empty
    ;

switch_statement:
        SWITCH '(' expression ')' '{' case_statement_list opt_default '}'

case_statement:
        CASE expression ':' commands
```

```
case_statement_list:
        case_statement_list case_statement
    |   case_statement
    ;

opt_default:
        DEFAULT ':' commands
    |   %empty
    ;

// ------------------------------ Loops ------------------------------

do_loop:
        DO '{' commands_with_break '}' WHILE '(' bool_expression ')'
    ;

for_loop:
        FOR '(' variable_declaration_with_assign ';' comparison_expression ';'
assign_command_no_semicolon ')' '{' commands_with_break '}'


// ------------------------------ Expressions ------------------------------

expression:
        numerical_expression
    |   bool_expression
    |   CHAR_LITERAL
    |   STRING_LITERAL
    |   object_instation
    |   identifier '(' method_call_parameter_list ')'  // Κλήση μεθόδου
          {
             const auto opt_error = parser_helper->getMethod(parser_helper->in_class.back(), $1);
             if(opt_error){
                yyerror(opt_error.value().c_str());
                YYERROR;
             }
          }
    ;


object_instation:
        NEW CLASS_NAME '(' ')'  // Δημιουργία αντικειμένου κλάσης
          {
             const auto opt_error = parser_helper->getClass($2);
             if(opt_error){
                yyerror(opt_error.value().c_str());
```

```
                YYERROR;
            }
        }
    ;

numerical_expression:
        INT_LITERAL
    |   double_literal
    |   identifier
    |   class_member
    |   '(' numerical_expression ')'
    |   '-' numerical_expression
    |   numerical_expression '+' numerical_expression
    |   numerical_expression '-' numerical_expression
    |   numerical_expression '*' numerical_expression
    |   numerical_expression '/' numerical_expression
    ;

bool_expression:
        bool_literal
    |   comparison_expression
    |   '(' bool_expression ')'
    |   bool_expression AND bool_expression
    |   bool_expression OR bool_expression
    ;

comparison_expression:
        '(' comparison_expression ')'
    |   numerical_expression '>' numerical_expression
    |   numerical_expression '<' numerical_expression
    |   numerical_expression EQ numerical_expression
    |   numerical_expression NEQ numerical_expression
    ;

bool_literal:
        TRUE    { $$ = true; }
    |   FALSE    { $$ = false; }
    ;

double_literal:
        DOUBLE_LITERAL DOUBLE_SP  { $$ = $1; }
    ;


// ------------------------------ Misc ------------------------------
opt_return:
```

```
        RETURN expression ';'
    |   RETURN ';'
    |   %empty
    ;

identifier:
        CLASS_NAME  { $$ = $1; }
    |   IDENTIFIER  { $$ = $1; }
    ;

data_type:
        INT     { temp_data_type = DataType::INT_TYPE; }
    |   DOUBLE  { temp_data_type = DataType::DOUBLE_TYPE; }
    |   BOOLEAN { temp_data_type = DataType::BOOL_TYPE; }
    |   CHAR    { temp_data_type = DataType::CHAR_TYPE; }
    |   STRING  { temp_data_type = DataType::STRING_TYPE; }
    ;

access_modifier:
        PUBLIC      { $$ = true; temp_access_modifier = true; }
    |   PRIVATE     { $$ = false; temp_access_modifier = false; }
    |   %empty      { $$ = true; temp_access_modifier = true; }
    ;

%%

int yyerror(const char *s) {
    std::cerr << "Error at line " << yylineno << " : " << s << std::endl;
    return 0;
}
```

parser_helper: προστέθηκε για να διαχειρίζεται οργανωμένα τη σημασιολογική ανάλυση του κώδικα, υποστηρίζοντας τη διαχείριση κλάσεων, μεταβλητών και μεθόδων.

Main: η συνάρτηση main λειτουργεί ως το σημείο εκκίνησης του προγράμματος. Ελέγχει αν έχει δοθεί σωστά το αρχείο εισόδου, το ανοίγει και το συνδέει με τον lexer για την ανάγνωση των δεδομένων. Στη συνέχεια, καλεί τον parser για να επεξεργαστεί τη γραμματική και εμφανίζει τα κατάλληλα

μηνύματα επιτυχίας ή αποτυχίας, διασφαλίζοντας μια ομαλή ροή από την εισαγωγή μέχρι την έξοδο.

CmakeLists: Το αρχείο CMakeLists.txt είναι υπεύθυνο για τη ρύθμιση και αυτοματοποίηση της διαδικασίας κατασκευής του προγράμματος. Συγκεκριμένα, διαχειρίζεται τη μεταγλώττιση του αρχείου flex_lexer.l από το Flex, παράγοντας το αρχείο flex_lexer.yy.cpp, και του αρχείου bison_parser.y από το Bison, παράγοντας τα αρχεία bison_parser.tab.cpp και bison_parser.tab.hpp. Διασφαλίζει τη σωστή συνεργασία μεταξύ Flex και Bison μέσω της δήλωσης εξαρτήσεων, ενώ συμπεριλαμβάνει τα παραγόμενα αρχεία στον φάκελο generated και τα ενσωματώνει στη διαδικασία κατασκευής. Επιπλέον, δημιουργεί το εκτελέσιμο με όνομα parser, συνδυάζοντας αρχεία όπως το main.cpp, το parser_helper.h, και τα αρχεία που παράγονται από Flex και Bison. Τέλος, συνδέει τις βιβλιοθήκες του Flex και του Bison με το εκτελέσιμο και εξασφαλίζει ότι τα απαραίτητα headers είναι διαθέσιμα, καθιστώντας τη διαδικασία κατασκευής πλήρως αυτοματοποιημένη και εύκολη στη διαχείριση.

2)

BNF2:<variable_declaration> ::= <access_modifier> <data_type> <identifier> <opt_more_variables> ";"
| <variable_declaration_with_assign>
<opt_more_variables_with_assign> ";"

<variable_declaration_with_assign> ::= <access_modifier> <data_type> <identifier> "=" <expression>

<opt_more_variables> ::= "," <identifier> <opt_more_variables>
| ε

<opt_more_variables_with_assign> ::= "," <identifier> "=" <expression> <opt_more_variables_with_assign>
| ε

BISON:variable_declaration:
access_modifier data_type identifier opt_more_variables ';'
{
parser_helper->addVariable($3, temp_data_type, $1);
}
| variable_declaration_with_assign opt_more_variables_with_assign ';'
;

variable_declaration_with_assign:
access_modifier data_type identifier '=' expression
{
parser_helper->addVariable($3, temp_data_type, $1);
}
;

opt_more_variables:
',' identifier opt_more_variables { parser_helper->addVariable($2, temp_data_type, temp_access_modifier); }
| %empty
;

opt_more_variables_with_assign:
',' identifier '=' expression opt_more_variables_with_assign { parser_helper->addVariable($2, temp_data_type, temp_access_modifier); }
| %empty
;

Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

3α) Στον κώδικα έχουν υλοποιηθεί οι απαραίτητες λειτουργίες για τον έλεγχο δήλωσης μεταβλητών και την επικύρωση κλήσεων μεθόδων. Αυτές διασφαλίζονται μέσω των συναρτήσεων getVariable και getMethod του αρχείου parser_helper.h.

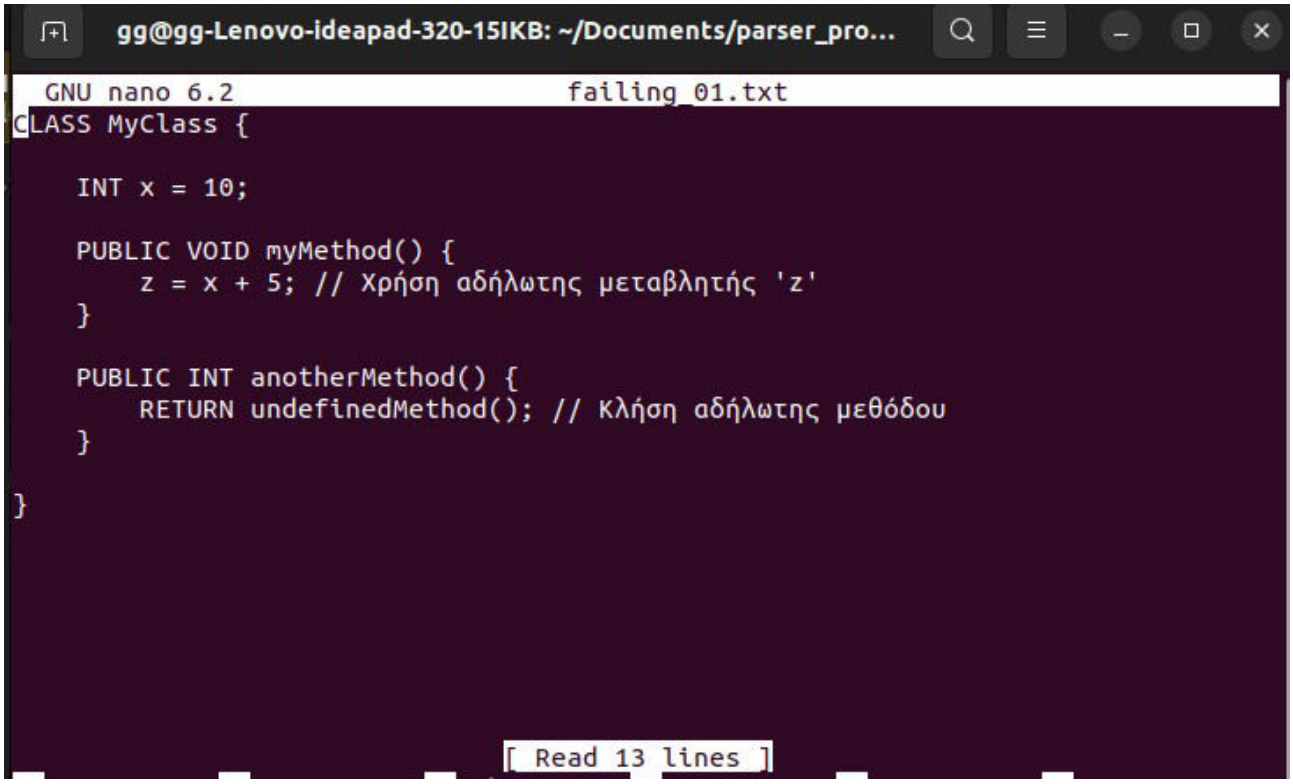
Έλεγχος Δήλωσης Μεταβλητών:
Κάθε φορά που γίνεται χρήση μιας μεταβλητής, ο parser καλεί τη **getVariable** για να ελέγξει:
1.Αν η μεταβλητή είναι δηλωμένη στην κλάση στην οποία ανήκει.
2.Αν η πρόσβαση στη μεταβλητή είναι επιτρεπτή (δημόσια ή ιδιωτική).

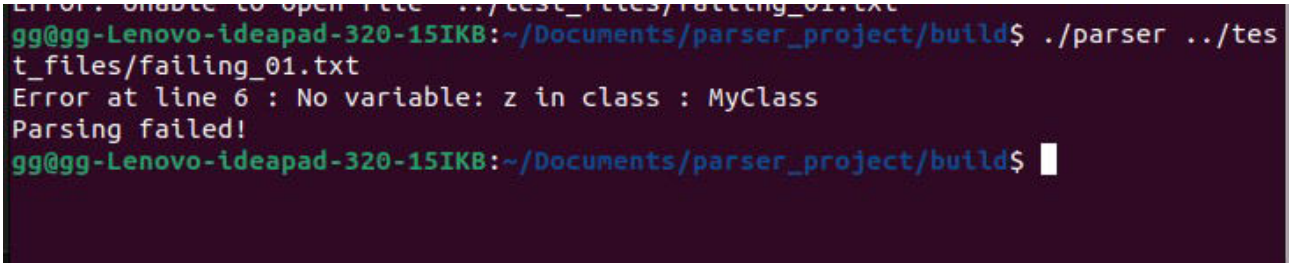
Έλεγχος Ορισμού Μεθόδων:
Ομοίως, κάθε φορά που γίνεται κλήση μιας μεθόδου, ο parser καλεί τη **getMethod** για να ελέγξει:
1.Αν η μέθοδος είναι δηλωμένη στην κλάση στην οποία ανήκει.
2.Αν η πρόσβαση στη μέθοδο είναι επιτρεπτή (δημόσια ή ιδιωτική).

Οι μέθοδοι getVariable και getMethod βασίζονται στη δομή δεδομένων **parsed_classes_map** για να εντοπίσουν αν μια μεταβλητή ή μέθοδος ανήκει σε μια συγκεκριμένη κλάση.
Η αναζήτηση γίνεται μέσω της χαρτογράφησης (map), όπου το κλειδί είναι το όνομα της κλάσης.

Διακοπή Ανάλυσης:
Όταν εντοπιστεί κάποιο σφάλμα στη δήλωση μιας μεταβλητής ή μεθόδου, η διαδικασία ανάλυσης διακόπτεται με χρήση της **YYERROR** και εμφανίζεται σχετικό μήνυμα μέσω της yyerror.

Failing_01





Στη μέθοδο myMethod, η μεταβλητή z χρησιμοποιείται χωρίς να έχει δηλωθεί μέσα στην κλάση ή στη μέθοδο.
Ο parser θα ελέγξει αν η z υπάρχει στη λίστα των δηλωμένων μεταβλητών της κλάσης MyClass μέσω της getVariable και θα επιστρέψει σφάλμα

3β) Έλεγχος Εμβέλειας Μεταβλητών:

Η **getVariable** ελέγχει:
1.Αν η μεταβλητή ανήκει στην τρέχουσα κλάση (χρησιμοποιώντας τη λίστα **in_class**).

2.Αν η μεταβλητή είναι δημόσια (public) ή αν η πρόσβαση σε ιδιωτική μεταβλητή (private) γίνεται από μη επιτρεπτή κλάση.
Αν εντοπιστεί προσπάθεια πρόσβασης σε ιδιωτική μεταβλητή από άλλη κλάση, εμφανίζεται μήνυμα σφάλματος.

Έλεγχος Εμβέλειας Μεθόδων:
Η **getMethod** λειτουργεί αντίστοιχα για τις μεθόδους. Αν μια μέθοδος είναι ιδιωτική (private) και καλείται από διαφορετική κλάση, εμφανίζεται σφάλμα.

Ο έλεγχος εμβέλειας αφορά μόνο μεταβλητές και μεθόδους που ανήκουν σε κλάσεις.

Failing_02

```
 GNU nano 6.2                    failing_02.txt
CLASS FistClass {

    PRIVATE INT myMethod(){
        RETURN 5;
    }

}

PUBLIC CLASS MyClass{

    INT x = FistClass.myMethod():

}

                    [ Read 14 lines ]
```

```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/parser_project/build$ ./parser ../tes
t_files/failing_02.txt
Error at line 11 : Cannot access: myMethod of class: FistClass because it's priv
ate
Parsing failed!
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/parser_project/build$
```

Στο παράδειγμα γίνεται μια προσπάθεια κλήσης της ιδιωτικής (PRIVATE) μεθόδου myMethod() της κλάσης FistClass από την κλάση MyClass.
Σύμφωνα με τους κανόνες πρόσβασης που έχουν εφαρμοστεί, μια ιδιωτική μέθοδος είναι προσβάσιμη μόνο εντός της κλάσης στην οποία ανήκει. Στην περίπτωση αυτή, το FistClass.myMethod() δεν είναι προσβάσιμο από την MyClass.

Για είσοδο το test_file: test_working_input.txt, έχουμε σωστά τα εξής αποτελέσματα