Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Η/Υ

Εργαστηριακή Άσκηση:

Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών

Συγγραφέας:

Μενεγάτος ΄Άγγελος,

Φλουρής Παντελής,

Κολτσάκης Χρυσάφης και

Υπεύθυνος Καθηγητής:

Ι. Γαροφαλάκης, Σ. Σιούτας,

Π. Χατζηδούκας

Αμαξόπουλος Γιώργος

Η εργασία κατατέθηκε για το μάθημα:

Αρχές Γλωσσών Προγραμματισμού Μεταφραστών

8 Σεπτεμβρίου 2024

Περιεχόμενα

Περιεχόμενα

Στοιχεία των μελών της ομάδας

Μενεγάτος ΄Αγγελος 1093426 3ο έτος up1093426@ac.upatras.gr

Φλουρης Παντελής 1093507 3ο έτος 1093507@ac.upatras.gr

Κολτσάκης Χρυσάφης 1084671 4ο έτος 1084671@ac.upatras.gr

Αμαξόπουλος Γιώργος 1093311 3ο έτος 1093311@ac.upatras.gr

Περιγραφή της γραμματικής της γλώσσας σε BNF

<small_letter> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m"

| "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

<capital_letter> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M"

| "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

<letter> ::= <small_letter> | <capital_letter>

<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<special_character> ::= "!" | "#" | "$" | "%" | "&" | "(" | ")" | "*" | "+" | "," | "-" | "." | "/" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "[" | "]" | "^" | "_" | "`" | "{" | "|" | "}" | "~"

<escape_sequence> ::= "\n" | "\t" | "\r" | "\b" | "\f" | "\"" | "\'" | "\\"

<int> ::= <digit> <int> | <digit>

<char> ::= "'" (<letter> | <special_character> | <escape_sequence>) "'"

<double> ::= <int> "." <int> "d"

<boolean> ::= "true" | "false"

<string> ::= '"' <string_characters> '"'

<string_characters> ::= <string_character> <string_characters> | e

<string_character> ::= <letter> | <digit> | <special_character> | <escape_sequence> | " "

<program> ::= <class_list>

<class_list> ::= <class_list> <class> | <class>

<class> ::= 'public' 'class' <class_name> '{' <class_body> '}'

<class_body> ::= <variable_declaration_list> <method_declaration>

| <variable_declaration_list>

| <methods>

| <variable_declaration_list> <methods> <class>

| <variable_declaration_list> <class>

| <methods> <class>

<class_name> ::= <capital_letter> <class_identity>

<class_identity> ::= <letter> <class_identity> | <digit> <class_identity> | '_' <class_identity> | e

<variable_declaration_list> ::= <variable_declaration> | <variable_declaration_list> <variable_declaration>

<variable_declaration> ::= <modifier> <return_type> <identifier> ';'

| <return_type> <identifier> ';'

<data_type> ::= "int" | "char" | "String" | "double" | "boolean"

<modifier> ::= "public" | "private"

<secondary_modifier> ::= "static" | "abstract" | "final" | "native" | "synchronized"

<identifier> ::= <letter_or_underscore> <identifier_body>

<letter_or_underscore> ::= <letter> | "_"

<identifier_body> ::= (<letter> | <digit> | "_") <identifier_body> | e

<method_access> ::= <identifier> "." <identifier>

<methods> ::= <method_declaration> | <methods> <method_declaration>

<method_declaration> ::= <modifier> <secondary_modifier> <return_type> <identifier> "(" <parameter_list> ")" "{" <method_body> "}"

<modifier> <return_type> <identifer> '(' <parameter_list> ')' '{' <method_body '}'

<modifier> <identifer> '(' <parameter_list> ')' '{' <method_body '}'

<return_type> ::= <data_type> | <class_name> | "void" | e

<parameter_list> ::= <data_type> <identifier> | <parameter_list> "," <data_type> <identifier> | e

<method_body> ::= <variable_declaration_list> <commands> | <commands>

<commands> ::= <commands> <command> | e

<command> ::= <assignment> | <loop> | <control> | <print> | <return> | <break>

<assignment>::= <identifier> '=' <expression> ';' | <identifier> '=' <object_creation> ';'

<object_creation>::= "new" class_name "()"

<literal>::= <int> | <char> | <String> | <boolean> | <double>

<expression> ::= <literal> | <suntheti_parastash>

<suntheti_parastash> ::= <addition> | <identifier> | <multiplication> | <division> | <subtraction> | '(' <expression> ')' | <call_method> | <method_access>

<addition> ::= <expression> '+' <expression>

<multiplication> ::= <expression> '*' <expression>

<division> ::= <expression> '/' <expression>

<subtraction> ::= <expression> '-' <expression>

<call_method> ::= <identifier> '(' <arguement_list> ')' | <identifier> '(' ')'

<arguement_list> ::= <identifier> | <arguement_list> ',' <literal> | <arguement_list> ',' <identifier> | <literal>

<loop> ::= <while> | <for>

<while> ::= "do" '{' <variable_declaration_list> <commands> '}' "while" '(' <condition> ')' ';'

| "do" '{' <commands> '}' "while" '(' <condition> ')' ';'

<condition> ::= <expression> <conop> <expression> | <expression>

<conop> ::= "||" | "&&" | "==" | "!=" | ">" | ">=" | "<" | "<="

<for> ::= "for" '(' <exp1> ';' <exp2> ';' <exp3> ')' '{' <variable_declaration_list> <commands> '}'

| "for" '(' <exp1> ';' <exp2> ';' <exp3> ')' '{' <commands> '}'

<exp1> ::= <data_type> <identifier> '=' <literal> | <identifier> '=' <literal>

<exp2> ::= <identifier> <conop> <literal>

<exp3> ::= <identifier> '+' <expression>

| <identifier> '-' <expression>

| <identifier> '/' <expression>

| <identifier> '*' <expression>

<control> ::= <if> | <switch_statement>

<if> ::= "if" '(' <condition> ')' '{' <variable_declaration_list> <commands> '}' <elseif> <else>

| "if" '(' <condition> ')' '{' <commands> '}' <elseif> <else>

<esleif> ::= <elseif> "else if" '(' <condition> ')' '{' <variable_declaration_list> <commands> '}' | e

| <elseif> "else if" '(' <condition> ')' '{' <commands> '}' | e

<else> ::= "else" '{' <variable_declaration_list> <commands> '}'

| "else" '{' <commands> '}'

<switch_statement> ::= "switch" '(' <identifier> ')' '{' <case> <default_opt> '}'

<case> ::= <case> "case" <literal> ':' <variable_declaration_list> <commands> <break> | e

| <case> "case" <literal> ':' <commands> <break> | e

<default_opt> ::= "default" ':' <variable_declaration_list> <commands>

| "default" ':' <commands>

<print> ::= "out.print" '(' <String> ')' ';' | "out.print" '(' <String> <ident_list>')' ';'

<list> ::= ',' <expression> | <ident_list> ',' <expression>

<return> ::= "return" <expression> ';'

<break> ::= "break" ';'

Τελικά αρχεία περιγραφής της γλώσσας

FLEX

```
%{
   #include "parser.tab.h"


   char str_buf[256];
   void yyerror(const char* err);
   int line_number = 1;
%}


%option noyywrap


INT          [0-9]+


LETTER       [a-zA-Z]
IDENT        ({LETTER}|_)({LETTER}|{INT}|_)*
```

```
%%

\n        { line_number++; }

"+"       { return PLUS; }
"-"       { return MINUS; }
"*"       { return TIMES; }
"/"       { return DIV; }

";"       { return SEMICOLON; }
"."       { return DOT; }
","       { return COMMA; }
":"       { return COLON; }
"int"     { return DATATYPE; }
"char"    { return DATATYPE; }
"double"  { return DATATYPE; }
"boolean" { return DATATYPE; }
"String"  { return DATATYPE; }

"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; }

"private"  { return PRIVATE; }
"public"   { return PUBLIC; }

"static"   { return STATIC; }
"abstract" { return ABSTRACT; }
"final"    { return FINAL; }
"native"   { return NATIVE; }
"synchronized" { return SYNCHRONIZED; }

"//".*              { /* Ignore single-line comments */ }
"/*"([^*]|\*+[^*/])*"*"+"/"  { /* Ignore multi-line comments */ }

[A-Z][a-zA-Z0-9_]*          { return CLASS_NAME; }

"class"                     { return CLASS; }
"{"                         { return LCURLY; }
"}"                         { return RCURLY; }

{INT}                       { return INT; }

\'([^\\]|\\[ntbrf'\\])\'      { return CHAR; }

{INT}"."{INT}"d"            { return DOUBLE; }
```

```
"true"                  { return BOOL; }
"false"                 { return BOOL; }

\"(\\.|[^"])*\"         { return STRING; }

"="                     { return EQUALS; }
"("                     { return LPAR; }
")"                     { return RPAR; }

"||"                    { return OROP; }
"&&"                    { return ANDOP; }
"=="|"!="               { return EQUOP; }
">"|">="|"<"|"<="       { return RELOP; }

"out.print"             { return OUTPRINT; }
{IDENT}                 { return IDENT; }

[ \t]+                  { /* Ignore whitespace */ }
.                       { fprintf(stderr, "Unexpected character: %s at line %d\n", yytext,
yylineno); }
```

BISON

```
%{
  #include <stdbool.h>
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <stdbool.h>

  extern FILE *yyin;
  int yyparse();



  extern int yylex();
  extern int line_number;
  extern void yyerror(const char* err);
%}

%define parse.error verbose

%token DATATYPE NEW RETURN VOID IF ELSE WHILE DO FOR SWITCH CASE
DEFAULT BREAK
```

```
%token INT CHAR BOOL STRING DOUBLE PRIVATE PUBLIC STATIC ABSTRACT
FINAL NATIVE SYNCHRONIZED OUTPRINT

%left OROP

%left ANDOP

%left EQUOP

%left NOTOP

%left RELOP

%token IDENT LCURLY RCURLY CLASS CLASS_NAME LPAR RPAR COLON DOT
SEMICOLON EQUALS

%left PLUS MINUS DIV TIMES

%token COMMA


%%



program:

    class_list


class_list:

      class_list class

      | class



class:

      PUBLIC CLASS CLASS_NAME LCURLY class_body RCURLY


class_body:

    | variable_declaration_list methods

    | variable_declaration_list
```

```
    | methods
    | variable_declaration_list methods class
    | variable_declaration_list class
    | methods class


variable_declaration_list:
      variable_declaration
    | variable_declaration_list variable_declaration

return_type:
  DATATYPE
  | CLASS_NAME
  | VOID;

variable_declaration:
  modifier return_type IDENT SEMICOLON
  | return_type IDENT SEMICOLON

methods:
              method_declaration
             | methods method_declaration

method_declaration:
  modifier secondary_modifier return_type IDENT LPAR parameter_list RPAR
LCURLY method_body RCURLY
  | modifier return_type IDENT LPAR parameter_list RPAR LCURLY method_body
RCURLY;
  | modifier IDENT LPAR parameter_list RPAR LCURLY method_body RCURLY;
```

```
modifier:
      PUBLIC
     | PRIVATE

method_access:
              IDENT DOT IDENT

secondary_modifier:
                 STATIC
                | ABSTRACT
                | FINAL
                | NATIVE
                | SYNCHRONIZED

parameter_list:
           | DATATYPE IDENT
           | parameter_list COMMA DATATYPE IDENT

method_body:
           variable_declaration_list commands
          | commands


commands:
     | commands assignment
     | commands loop
     | commands control
```

```
    | commands print
    | commands return
    | commands break


assignment:
        IDENT EQUALS expression SEMICOLON
       |IDENT EQUALS object_creation SEMICOLON

object_creation:
        NEW CLASS_NAME LPAR RPAR


literal:
        INT
       | CHAR
       | STRING
       | BOOL
       | DOUBLE

expression:
        literal
       | suntheti_parastash

suntheti_parastash:
        addition
       | IDENT
       | multiplication
```

```
            | division
            | subtraction
            | LPAR expression RPAR
            | call_method
            | method_access

addition:
      expression PLUS expression
multiplication:
      expression TIMES expression
division:
      expression DIV expression
subtraction:
      expression MINUS expression

call_method:
             IDENT LPAR argument_list RPAR
            | IDENT LPAR RPAR

argument_list:
            IDENT
           | argument_list COMMA literal
           | argument_list COMMA IDENT
           | literal

loop:
                  while
                  |for
```

while:

DO LCURLY variable_declaration_list commands RCURLY WHILE LPAR condition RPAR SEMICOLON

| DO LCURLY commands RCURLY WHILE LPAR condition RPAR SEMICOLON

condition:

expression conop expression

| expression

;

conop:

RELOP

| EQUOP

| NOTOP

| ANDOP

| OROP

;

for:

FOR LPAR exp1 SEMICOLON exp2 SEMICOLON exp3 RPAR LCURLY variable_declaration_list commands RCURLY

| FOR LPAR exp1 SEMICOLON exp2 SEMICOLON exp3 RPAR LCURLY commands RCURLY

exp1:

DATATYPE IDENT EQUALS literal

| IDENT EQUALS literal

exp2:

IDENT conop literal

exp3:

IDENT PLUS expression

| IDENT MINUS expression

| IDENT DIV expression

| IDENT TIMES expression

control:

if

| switch_statement

if:

IF LPAR condition RPAR LCURLY variable_declaration_list commands RCURLY elseif else

| IF LPAR condition RPAR LCURLY commands RCURLY elseif else

elseif:

| elseif ELSE IF LPAR condition RPAR LCURLY variable_declaration_list commands RCURLY

| elseif ELSE IF LPAR condition RPAR LCURLY commands RCURLY

else:

| ELSE LCURLY variable_declaration_list commands RCURLY

| ELSE LCURLY commands RCURLY

```
switch_statement:
   SWITCH LPAR IDENT RPAR LCURLY case default_opt RCURLY
;

case:
   | case CASE literal COLON variable_declaration_list commands break
   | case CASE literal COLON commands break
;

default_opt:
   /* empty */
   | DEFAULT COLON variable_declaration_list commands
   | DEFAULT COLON commands
;

print:
                    OUTPRINT LPAR STRING RPAR SEMICOLON
                  | OUTPRINT LPAR STRING list RPAR SEMICOLON

list:
         COMMA expression
        | list COMMA expression

return:
                  RETURN expression SEMICOLON
```

```
break:
                BREAK SEMICOLON



%%
int flag =0;

void yyerror(const char *s) {
   printf("\n\n\nError: %s at line %d\n", s, line_number);
   printf("\nProgram terminated unsuccessfully.\n");
   flag = 1;
}

int main(int argc, char* argv[]) {

   int token;
   yyin = fopen(argv[1], "r");

   if (yyin == NULL) {
      printf("%s: File not found\n", argv[1]);
      return 1;
   }

   char ch;
   while ((ch = fgetc(yyin)) != EOF) {
      putchar(ch);
   }
```

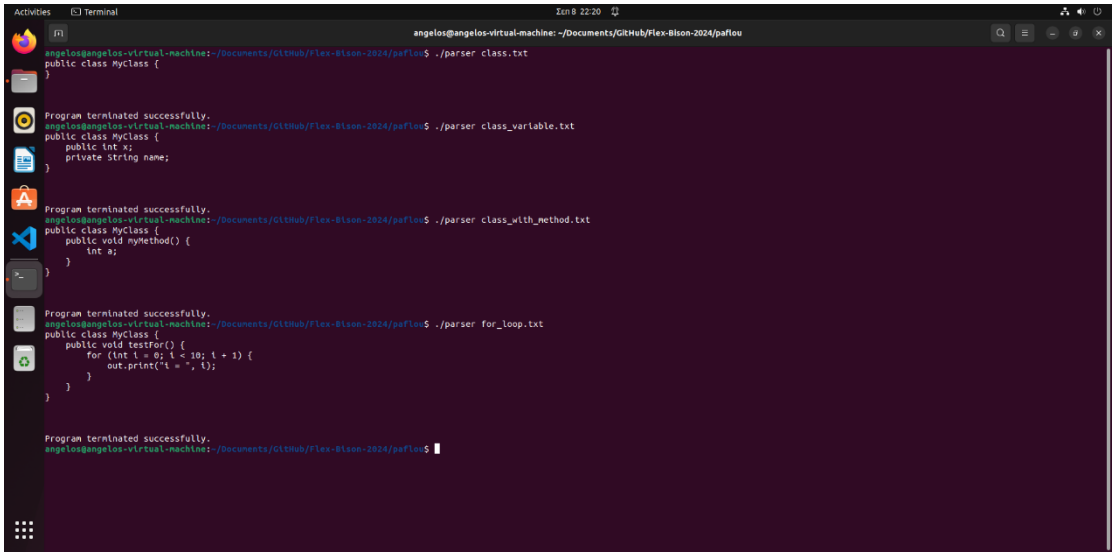
```
    rewind(yyin);

    yyparse();
    fclose(yyin);

    if(flag==0)
        printf("\n\n\nProgram terminated successfully.\n");
    return 0;
}
```

Screenshots παραδειγμάτων εφαρμογής

Παραδείγματα ορθής λειτουργίας



```
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser if_else.txt
public class MyClass {
    public void checkCondition() {
        if (x == 5) {
            out.print("x is 5");
        } else {
            out.print("x is not 5");
        }
    }
}


Program terminated successfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser method_call.txt
public class MyClass {
    public void callMethod() {
        myMethod();
    }

    public void myMethod() {
        out.print("Method called");
    }
}


Error: syntax error, unexpected LPAR, expecting EQUALS at line 3

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser object_creation.txt
public class MyClass {
    public void createObject() {
        MyClass obj;
        obj = new MyClass();
    }
}


Program terminated successfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$
```

```
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser switch.txt
public class MyClass {
    public void testSwitch(int value) {
        switch (value) {
            case 1:
                out.print("One");
                break;
            case 2:
                out.print("Two");
                break;
            default:
                out.print("Default");
        }
    }
}


Program terminated successfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser while_loop.txt
public class MyClass {
    public void testWhile() {
        do {
            x = x - 1;
        } while (x > 0);
    }
}


Program terminated successfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$
```

Παραδείγματα λανθασμένης λειτουργίας

```
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser class.txt
public flass MyClass {
}


Error: syntax error, unexpected IDENT, expecting CLASS at line 1

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser class_variable.txt
public class MyClass() {
    public int x;
    private String name;
}


Error: syntax error, unexpected LPAR, expecting LCURLY at line 1

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser class_with_method.txt
public void myMethod()
public class MyClass {
    {
        int a;
    }
}


Error: syntax error, unexpected VOID, expecting CLASS at line 1

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$
```

```
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser for_loop.txt
public class MyClass{{ {
    public void testFor() {
        for (int i = 0; i < 10; i + 1) {
            out.print("i = ", i);
        }
    }
}


Error: syntax error, unexpected LCURLY, expecting RCURLY at line 1

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser if_else.txt
public class MyClass {
    public void checkCondition() {
        if (if) {
            out.print("x is 5");
        } else {
            out.print("x is not 5");
        }
    }
}


Error: syntax error, unexpected IF at line 3

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser method_call.txt
public class MyClass {
    public void callMethod() {
        myMethod();
    }

    public void myMethod {
        out.print("Method called");
    }
}


Error: syntax error, unexpected LCURLY, expecting EQUALS at line 3

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$
```

```
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser object_creation.txt
public class MyClass {
    public void createObject() {
        MyClass obj;
        obj = new MyClass();

    }


Error: syntax error, unexpected end of file, expecting RCURLY at line 7

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser switch.txt
public class MyClass {
    public void testSwitch(int value) {
        switch (value) {
            case 1:
                out.print("One");
                break;
            case 2
                out.print("Two");
                break;
            default:
                out.print("Default");
        }
    }
}


Error: syntax error, unexpected OUTPRINT, expecting COLON at line 8

Program terminated unsuccessfully.
angelos@angelos-virtual-machine:~/Documents/GitHub/Flex-Bison-2024/paflou$ ./parser while_loop.txt
public class MyClass {
    public void testWhile() {
        while {
            x = x - 1;
        } while (x > 0);
    }
}


Error: syntax error, unexpected WHILE, expecting RCURLY at line 3

Program terminated unsuccessfully.
```

Σχόλια – Παραδοχές

Έχει υλοποιηθεί το πρώτο ζητούμενο 1α και 1β. Είναι λειτουργικό και δεν έχουμε εντοπίσει κάποιο λάθος στον κώδικα.

Οι εμφωλευμένες κλάσεις βρίσκονται αναγκαστικά στο τέλος της κλάσης που την καλεί (κλάση πατέρας).