

Project 1: Lexical Analyzer

Programming Language Name: Ghost

Tunahan Burak Dirlik

20170808076

Aim

Ghost progrramming language is based on Java language. Main aim of the Ghost is facilitate the syntax of java and code writing.

That's why many many reserved words defined again as abbreviated. But this language is imperarive so not oop base. We can use this language

for mathematical calculations, textual operaitons.

Reserved words:

JAVA GHOST

int int

double db

long l

char cc

boolean bn

true true

false false

for f

if if

else elif

while w

void v

array arr

public pb

private pr

main main

print show

scan sc

**Example program**

main(){

pb l factorial(10):

if(isprime(23)){show(“true”):

}

}

pb l factorial(int n){

l fact = 1;

f(int i=2: i<=n: i++){

fact=fact\*i:

}

show(fact):

return fact:

}

stt bn isprime(int n){

if(n<=1) return false:

f( int i =2: i<n: i++){

if(n %i==0){return false:}

return true:

}

}

BNF FORM

-- Operators --

<multipication\_operator> ::= \*

<division\_operator> ::= /

<addition\_operator> ::= + | -

<relational\_operator> ::= < | > | <= | >=

<comment> ::= ## <characters>

<character> ::= <upper\_case\_letter> | <lower\_case\_letter> | <digit> | ! | ' | ^ | # | + | - | % | & | / | ( | ) | = | ? | \* | < | > | - | \ | . | ,

<upper\_case\_letter> ::= A | B | C | … | Z | \_

<lower\_case\_letter> ::= a | b | c | … | z | \_

<digit> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<sign> ::= + | -

<type\_name> ::= int | db | l | cc | bn

<data\_type> ::= array

<int> ::= <sign> ? <number>

<array type> ::= <type><> [ ]

<boolean literal> ::= true | false

<method declaration> ::= <method header> <method body>

<method header> ::= <method modifiers>? <result type> <method declarator>

<result type> ::= <type> | v

<method modifiers> ::= <method modifier> | <method modifiers> <method modifier>

<method modifier> ::= pr| pb| stt

<method declarator> ::= <identifier> ( <formal parameter list>?

<method body> ::= <block>

<result type> ::= <type> | void

<if then statement>::= if ( <expression> ) <statement>

<else statement>::= elif <statement>

<selection-statement> ::= if ( <expression> ) <statement>

| if ( <expression> ) <statement> elif<statement>

<iteration-statement> ::= while ( <expression> ) <statement>

<while statement> ::= w( <expression> ) <statement>

<expression-statement> ::= {<expression>}? :

<statement> ::= | <expression-statement>

<expression-statement> ::= {<expression>}? :

<for statement> ::= f( <for init>? : <expression>? : <for update>? ) <statement>

<for init> ::= <statement expression list> | <local variable declaration>

<for update> ::= <statement expression list>

<statement expression list> ::= <statement expression> | <statement expression list> , <statement expression>

<return statement> ::= return <expression>? :

<variable declarators> ::= <variable declarator> | <variable declarators> , <variable declarator>

<variable declarator> ::= <variable declarator id> | <variable declarator id> = <variable initializer>

<variable declarator id> ::= <identifier> | <variable declarator id> [ ]

<variable initializer> ::= <expression> | <array initializer>

<empty statement> ::= :

<static initializer> ::= stt <block>

<scan\_statement> :: = sc <var\_name>

<priny\_statement> :: show <expression>