|-------------------------------------------------------------|

| CONTEXT FREE GRAMMAR |

|-------------------------------------------------------------|

S:

import java. packages ;

S | A;

A:

class name { B;

B:

public static void main ( String args [ ] ) { start | funct return id ; }

print :

System.out.println( “ str “ | + var\_list ) ;

input :

Scanner name = new Scanner ( System.in ) ; D | type id | type id input

D:

type id = name.methods ;

func:

access\_spec class name { A’ | A’

A’:

access\_spec static type name( input | lambda ) { stmt ; } func | B | lambda

6

stmt:

assign\_exp ; stmt | array ; stmt | unop | lambda

array:

type[ ] id = Value | type id[ ] = Value | type[ ] id = new type [ ] Value

Value:

{ num } Value

start:

for\_exp | while\_exp | if\_exp | decl\_exp | print

for\_exp:

for ( decl\_exp | exp ; condition ; exp | unop ) { stmt ; start }

while\_exp:

while(condition) { start ; | assign\_exp ; start ; | unop ; }

if\_exp:

if ( condition ) { start ; | assign\_exp ; } else { start ; | assign\_exp ; }

exp:

assign\_exp | lambda

condition:

E relational\_op E | E logical\_op E | Boolean | lambda

unop:

id ++ | ++ id | id -- | -- id

relational\_op:

> | < | >= | < = |! = | ==

logical\_op:

&& | || | !

Boolean:

true |false

7

E:

E + T | E - T | T

T:

T \* F | T / F | F

F:

F % G | G

G:

id | num

assign\_exp:

id = E

decl\_exp:

type assign\_exp ;

type:

boolean | char | byte | short | int | long | float | double | String

digit:

[ 0 – 9 ]

letter:

[ a – z A – Z ]

name:

letter +

id:

letter | \_ B’

B’ :

letter \_ | digit B’ | lambda

num:

digit +

str:

[ a – z A – Z 0 – 9 ] str | lambda

var\_list:

id var\_list | id

8

access\_spec:

public | private | protected

packages:

lang | io | util | util \* | applet |awt | net

methods:

nextInt | nextBoolean | nextByte | nextDouble | nextLine | nextLong | nextShort

Special Symbols and Operators :

<parenthesis ,( ( , ) , { , } , [ , ] ) >

<semicolon , ; >

<comma , , >

<asterisk , \* >

<unary\_op , ( ++ ,-- ) >

<arithematic\_op , ( + , - ,\* / , % ) >

<relop , ( < , > , <= , > = , ! = , == ) >

< logop , ( && , || ,! ) >