**VEĽKÉ** sú neterminály, **malými** sú terminály

**P** – program (zaciatocny symbol)

**C –** class

**AC –** after class

**ACD** – after class dalsie

**F** - funkcia

**EOF** – end of file (no shit Sherlock :O)

**AL** – arguments list (parametre funkcie)

**AD**  - arguments definiton (definicia arugmentu)

**ALE** – arguments list extended

**S** – statement (príkaz)

**E** – experession (výraz)

**DEFS** – definícia statickej premennej

**DEFC** – definícia lokálnej premennej

**I** – inicializácia

**VAL**– argumenty volanej funckie

**VAL1** – rozsirenie argumentov

**VS** – vysledna hodnota, detto priradenia

**idc –** identifikator classy

**idf –** identifikator funkcie

**id –** identifikator premennej

P -> C P ak nebude zatvorka skocit na succes // Za definovanou classou sa môže nachádzať ďalšia classa

P -> EOF // Za poslednou funckiou je už ale len EOF

C -> class idc { AC } // class má povinný terminál **class**

// za class môže nasledovať buď funkcia(e) alebo definícia statickej premennej

AC -> static type id ACD AC

ACD -> ( AL ) S // v pripade funkcie

ACD -> ; // deklaracie statickej premennej

ACD -> = E; S // inicializacie statickej premennej

AC -> F AC // za Classou je definovaná aspoň 1 funkcia

AC -> DEFS AC // globalna (staticka) premenna môže byť dekl. i za funkciou

AC -> ε

I -> = E ; // v prípade že sa inicializuje, tak uložíme vzhodnotený výraz

I -> ; // ak je len deklarácia, inicializácia sa vynecháva

DEFC -> type id I

F -> static type idf ( AL ) S // type je návratový typ funkcie, id jej identifikátor

AL -> AD ALE

AL -> ε // v prípade že je argument len jeden

ALE -> , AD ALE // oddelíme čiarkou a nasleduje ďalší argument i s deklaráciou

ALE -> ε

AD -> type id

S -> { S } // výraz to složený výraz

S -> if ( E ) S else S // vetva if, zatiaľ bez rozšírenia, teda nutný else

S -> while ( E ) S

S -> return E ;

S -> DEFC ; S

S -> id = VS ; // do premennej s identifikatorom id priradi vysledna hodnota

S -> VS ;

S -> ε

VS -> E ;

VS -> idf( VAL ) ; S // argumenty volanej funckie

VAL -> ε // volame funkciu bey argumentov

VAL -> E VAL1

VAL1 -> , E VAL1

VAL1 -> ε

static declare = declaration unpack

after class = class unpack

stat\_declare\_after = method declaration unpack

params = parameters

declaration = declare

type = data type

params\_after = parameters\_unpack

main\_body = method\_unpack

next\_param = next\_parameter

st\_list = statement\_list

expr = expression

var\_declare = var\_value

func\_params = function\_call\_param

in\_body = inner bodz

extension = addition