Sintaksna analiza – cilj nam je da na osnvou cupa dobijemo sintaksni analizator.

Radio nesto o logovanju,sad prelazi na sintaksni analizator. Otvorimo specifikaciju I vidimo sta program ima – ima listu deklaracija promenljivih I metoda,jer su neterminali u ebnf notaciji izmedju viticastih zagrada sto oznacava ponavljanje proizvoljan broj puta.nas zadatak da prevedemo ovu notaciju u notaciju koju zahteva cup format. Pravimo novi fajl koji se zove mjparser.cup .pokusavamo da transformisemo prvu smenu. Program = “program” ident {vardecl} … dakle imamo terminale I neterminale. U cup fajlu mora da ima neki nacin definisanja terminala I neterminala. Cup fajl se sastoji od vise sekcija.prva importi,druga akcije,treca sam cup gde imamo na pocetku definisanje terminal I neterminala.ne pisemo ih redom odmah,mozemo odmah trecu.evo pisemo jednu smenu.NETERMINALE PISEM VELIKIM POCETNIM SLOVIMA, ::= se odvaja leva od desne strane,terminalni simboli svim velikim slovima. Zavrsava se smena sa ;.

Napisao smenu Program::=PROG IDENT VarDecList LBRACE MehtodDecList RBRACE;

Sad definisemo terminale iznad

terminal PROG,IDENT,LBRACE,RBRACE;

I neterminale

non terminal Program,varDecList,MethodDecList.

Tako sad radimo za sve smene.

VarDecList ::= VarDecList VarDecl | ; ovo izmedju je prazna smena (prikazivanje nula ili vise primenljivih liste) – primenjujemo levu rekurziju ili desnu rekurziju.kada treba da napisemo alternativnu,za oznacanje ili koristi se pipe( | ) koji moze bilo gde ali Koja ga pise ispod smene. Komentari se pisu pomocu /\* \*/ obrati paznju na tacku zarez,cesto izaziva greske,pise se na kraju smene . |.

VarDecl ::= Type IDENT SEMI;

DAKLE KADA IMAM PONAVLJANJE { } JA KRAIRAM NOVI NETERMINAL KOJI JE USTVARI LISTA TIH TERMINALA,A TO RADIM POMOCU REKURZIJE.

PRIMER

StatementList ::= StatementList Statement

|

/\* epsilon \*/

;

KAD PISEM NETERMINALE NE PISEM non terminal I nevedem ih sve u vise redova

Nego na pocetku svakog novog reda pisi non terminal

Non terminal bla bla

Non terminal blab la blab al

A ne

Non terminal bla bla

Blab la blab al.

Ona klasa sym koju smo sami pisali juce ce biti sama izgenerisana.samo treba u cup fajlu isto da nazivamo simbole da bi on stvorio iste :D

Kad imamo najmanje jednu vrednost a najvise proizvoljno odvojenih zarezom-

formparamlist = param | formparamlist,param

ZNACI RAZLIKUJEMO KAD IMAMO 0 ILI VISE VREDNOSTI I NAJMANJE 1 ILI VISE VREDNOSTI

Kad smo napisali sve smene I definisali sve non terminals I terminals pravimo port sekcije

Prvo pisemo paket

Package rs.ac.bg.etf.pp1;

Import java\_cup.runtime.\*;

Import java.io.\*;

Import org.apache.log4.\*;

Mi smo na casovima pricali o oporavku od sintasknih gresaka.Parser moze prvo pokusati da se oporavi od greske ako ugradimo u sintaksu smene za oporavak od greske. Ako nemamo te smene onda je nastala greska od koje ne moze da se oporavi.tako da parser prijavljuje sve greske kroz sistem metoda koje sluze za tu namenu I zovu se printrecovered ili tako nekako error report (ne zna on kako se zove),a ako ne moze da se oporavi onda poziva funkciju koja ce da ispise fatal error I zaustavlja parsiranje.

Zbog toga cemo preklopiti te metode koje parser poziva kada prijavljuje sintaske greske,da vidimo kako ce to sada da izgleda.

Od ovog parsera nastaje fajl sa dve klase -parser I action koja sprovodi akcije.ceo kod od parsera ide u parser,tu ce cup izgenerisati promenu smena.a mi mozemo dodati I svoj utility code tu ili user defined code. To mozemo da uradimo tako sto cemo u parser dodati sekciju koja se zove parser code

Dakle ispod ovih importa sto sam pisao unesem

Parser code {:

Ovde pisem metode,ako kaze nesto jos napisacu ih ja.ali on ih je vec pripremio pa samo radi copy paste.

Kopiraj metode ovde sa vezbe. Iz atributa left dobijamo informaciju o liniji kod metode za ispisvanje greske.brojanje kolona smo ukljucili jos u lekseru.

:}

Scan with {:

:}

Scan with sluzi za presretanje tokena koje salje leksicki analizator I onda rad parsera.parser ce da dobije lekser sa kojim radi,od njega ce da trazi next token,I njega ispisuje.

Sada pitanje kako cemo da generisemo kod.otvorimo komandnu liniju,pozicioniram se u projekat,I opet se pozicioniram u lib.tu je smestena cup generator parsera biblioteka.

Komanda za kreiranje :

java -jar java-cup-lia.jar -destdir ..\src\rs\ac\bg\etf\pp1 – parser MJParser ...\spec\mjparser.cup

Sym klasu preimenujemo u symold samo da bi videli sta ce da se desi.

Kada pokrenem ovu komandu iz komandne linije iznad kreirace se nova klasa sym :D

Sve ovo moze I iz run configuration main je main iz java cup a argumenti su

-destdir src\rs\ac\bg\etf\pp1 -parser MJParser spec\mjparser.cup

Ako za neki terminal zelimo da definisemo tip da ne bi radili konverziju posle stavimo Integer ispred.

terminal Integer NUMBER

terminal String IDENT

upozorenja dobijamo kada ne mozemo da dodjemo do svih neterminala od startnog simbola.

Na samom kraju mozemo vidi kako se radi prebrojavanje deklaracija,definicija I drugih sintaksnih gresaka.

Sta nam treba u gramatici,treba nam akcija,nekakva redukcija.odnosno da taj dogadjaj presretnemo I ugradimo svoj kod,unosimo u gramatiku akcije tako da bude kao translaciona.

U smeni na kraju desne strane smene dodajemo akciju pomocu {: :} gde treba da inkrementiram nekakav cnt, npr printCallCount++;

Definisemo je u onom utility kodu print code :{ }: ili kako vec.

OSTALO MI JE ZADNJIH 20 MINUTA DA ODGLEDAM,OPORAVAK OD GRESKE