Snimak 1 – Leksikca analiza

Prilikom kreiranja novog projekta nama je potreban obican java projekat.Sledeci korak wizard nam daje mogucnost imenovanja projekat.Zvace se MJCompiler.Sledece podesavanje je podesavanje java virtuelne masine.Ovde je ponudjena eclipseova verzija izvrsnog okruzenja.Ja cu dozvoliti da se koristi moje okruzenje ,a to je java7. Ako zelite da dodate vasa okruzenja kliknite an configure java envirements.Otvara se dijalog,klikenemo na add,directory I odabrati folder u kome je instalirana druga java virtuelna masina.ja sam ovde instalirao I jdk I virtuelnu masinu.TO SAM SADA URADIO CISTO RADI ILUSTRACIJE,A ZADOVOLJICU SE VEC PREDLOZENIM JAVA IZVRSNIM OKRUZENJEM.i na samom kraju idemo na poslednji prozor wizard.u poslednjem prozoru wizard mozemo da dobijemo preview nekakve structure projekta koju ce eclipse napraviti kada kliknemo na dugme finis.on je ponudio folder src u okviru java projekta I on je namenjen za smestanje java izvornog koda.ukoliko zelite mozete da dodajete nove foldere.mi smo sada recimo zadovoljni I kliknucemo na dugme finis a onda cu da dodam nove direktorijume. Na pitanje da li zelimo da podesimo sa perspektivom nesto.Medjutim ovo sto sam dobio sada nije krajnji oblik koji sam zeleo jer zelimda dobijem dekompoziciju svog projekta.

U SRC direktorijum ide kod koji pisemo mi rucno ili kod koji je izgenerisan pomocu generator sintaksnih I leksickih analizator.da vidimo sta nam jos treba u projektu.treba nam implementacija leksera I parsera Iz specifikacije.pa da ne bi mesali kod I specifikacije pravim novi src folder koji cu nazvati spec.implementacija leksera I parsera bice zasnovana na standardnim bibliotekama.zbog toga nam treba I folder u koji cemo staviti sve biblioteke koje koristimo.nazovimo ga lib,a onda u njega stavljamo sve moguce biblioteke koje cemo koristiti.ako napravimo vec u jednoj fazi leks analizator bice dobro da ga testiramo,pa pravimo src folder u koji cemo da pisemo testovi.dakle razdvajamo testove od korisnog koda.ako budemo koristili eksterne biblioteke,lako se moze desiti da ce biti potrebna I njihova konfiguracija.kreiramo novi folder koji se naziva config.sadrzi sve fajlove koji se ikada mogu pojaviti.imamo strukturu koja zadovoljava opste uslove za razvoj.

Prvo diskutujemo o smestanju izvornog koda,odnosno odredjujemo pakete u koje smestamo izvorni kod.Paket za izvorni kod nazvacemo kao internet domen za pp1 ali u obrnutnom redosledu.

Testovi ne smeju da budu u neimenovanom paketu u test folderu jer ne mozemo da ih importujemo.zbog toga stavljamo testove u paket,isto ime kao I src folder pa ne mora da se importuje.

PRVI ZADATAK:

Identifikacija svih tokena I vracanje jedinstvenih identifikatora za sve tokene.Nama ce generator parsera izgenerisati ovu klasu (sym.java),ali dok je ne izgenerise moracemo rucno mi da je pisemo pa kad dodjemo do sintaksne Analize OBRISACEMO JE ,JER NAM SAMO SAD SLUZI ZA RAZVOJ,PARSER GENERISE PRAVILNU.

Kada napravimo sym.java ,ono sto treba pisati jeste SPECIFIKACIJA LEKSERA.Zato u folderu specs dodajemo fajl KOJI NAZIVAMO MJLEXER.FLEX.Sada se otvorio prozoor za pisanje fajla,KOJI IMA TRI SEKCJIE.PRVA JE IMPORT SEKCIJA,DRUGA RAZDVOJENA OD PRVE SA %% je JFLEX DIREKTIVE I treca odvojena od prethodne sa %% znakom I ona sadrzi regularne izraze.Mi treba da ispisemo sve sekcije.generator generise specifiikaciju na jeziku java.ne proverava sintaksu I semantiko programa pa moramo da unesemo import iskaze da sprovedu ispravno prevodjenje koda leksickog analizatora.da vidimo sta nam je potrebno u sekciji za importe.kao prvo zelimo da smestimo to u paket rs ac etf bg pp1 pa to pisemo u delu import.Nas analizator treba da bude CUP KOMPATIBILAN I implementira skener interfejs pa dodajemo direktivu kup u sekciji directive.zelimo da brojimo linije I kolone I to oogucavamo direktivama line I column.krajnji proizvod generisanja leksera iz ove specifikacije bice java klasa.mi smo u mogucnosti da u tu java klasu ubacimo usluzni kod.ukoliko zelimo da u tu java klasu upisemo neki kod moramo da ga upisemo ovako. Pocinje %{ I zavrsava se %} I tu pisemo usluzne metode.pravimo dve fabricke metode koje ce praviti tokene.da vidimo koje su to metode,nazvacemo Ih new symbol koja ce ga praviti na osnovu tipa,a druga na osnovu tipa I vrednosti.ovo su te metode koje ce praviti simbole. Vracace symbol koja se nalazi u cup paketu,pa onda import sekciju ukljucivanjem paketa symbol iz cup-a.nas lekser bice implementiran kao automat I kao takav imace jedno stanje u okviru kojeg vrsi citanja ulaznih sekvenci.potrebno nam je jos jedno dodatno stanje koje odgovara obradi komentara.u default stanju reaguje na svaki znak koji citai gleda da ga upari.medjutim kad smo u stanju komentar nije bitno sta zatice pa onda lekser treba drugacije u ovom stanju da se ponasa. Dodavanje novog stanja radi se direktivom xstate IME\_STANJA.kada lekser stigne do kraja ulaznog fajla moze da izvrsi akciju u tom zavrsnom trenutku.za specificiranje akcije moze se koristiti direktiva #eofval{ I #eofval} I mi cemo napisati da se u tom trenutku vrati eof symbol koji se ne bi mogao procitati iz ulaznog fajla.SADA PRELAZIMO NA PISANJE trece I poslednje sekcije iz specifikacije a to je sekcija regularnih izraza I omogucavamo odgovarajuce akcije kada okine neki od regularnih izraza.na samom pocetku zelimo da lekser preskace sve bele linije u prazne znakove u kodu.zato dodajemo akciju za njihovo ignorisenje u trecu sekciju.na njih se ne reaguje ni na kakav nacin.sada pisemo za nase lekseme iz klase sym.java.

Npr za nasu leksemu “program” pisemo akciju { return new\_symbol(sym.PROG,yytext());}

Gde je yytext metoda koja vraca sekvencu znakova koji su procitani.sym.Prog, je naziv kako smo mi definisali. OVE AKCIJE TREBA ZADAVATI U ODREDJENOM STANJU.(preskicoo ovo sa prethodne recenice kao ne treba. Sada obradjujemo komentare.

Kada naidjemo na // akcija nece biti vracanje nikakvog tokena nego stavljamo automat u stanje obrade komentara.Nakon toga sve znakove koje bude automat citao obradjivace se u stanju komentara.

. bez navodnika oznacava bilo koji znak.

Podrazumevano stanje je yyinitial.

<comment> . { yybegin(comment);}

<comment> “\r\n” { yybegin(yyinital); }

Sad pisemo regularni izraz za celobrojne konstante.okida u yyinitial stanju,akcija je vracanje simbola sa vrednosnim delom. Akcija je vracanje novi token,koji ima identifikator number,a vrednost procitani broj.

Sto se tice idnetifikatora mogu poceti slovom a onda se moze naci proizvoljan broj slova znakova.vraca se symbol sa identifikatorom ident a vrednost je procitani znak.

Jedan poseban regualran izraz na kraju jer je poslednji koji ce se proveriti I ako se zadovolji jeste ispis leksicke greske na liniji koja je upravo obradjena.

Ovim smo zavrsili formiranje specifikacije naseg leksera I to sada cuvamo.

Sada treba ukljuciti biblioteku koja ima klasu simbola I leksera kojeg generisemo.to su java cup I jflex.zatim cu da ih stavim na class path da moze da cita klase kompajler.(properties projekta,pa java build path pa libraries).

Generisanje klasa dva nacina.prvi preko komandne linije,pozicioniramo se u workspace,odemo u mjcompiler I sada udje u direktorijum lib,gde se nalaze biblioteke,

Dakle pozicioniramo se u C:\pp1domaci\workspace\MJCompiler\lib> I ukucamo :

“java -jar JFlex.jar -d ..\src\rs\ac\bg\etf\pp1 ..\spec\mjlexer.flex” I sada je izgenerisan nas Yylex.java lekser.

Odemo u projekat I osvezimo projekat I izgenerisan je yylex.

Drugi nacin :

Iz aplikacije odemo na run configuration pa new pa klasu koju cemo da pokrenemo,to je main Iz jflexa I dodajemo argumente komandne linije. Konfiguraciju nazovemo lexergenerator I argumenti su :  
“-d src\rs\ac\bg\etf\pp1 spec\mjlexer.flex”.Sada kada smo dodali argumente idemo na dugme apply I run da vidimo da li radi.ako kazemo run kreira se nova klasa

Napravio onaj glup test primer a sad prica log4j. dodao paket util pa je dodao class path preko properties kao I neku iznad klasu vec sam pisao.odvojio je folder config za nju . xml fajl ovaj sadrzi appendere,jedan za konzolu jedan za fajl,fajl koji je to je logs/mj.log.imamo root za logere,nece da duzi o tome.i da definisemo prioritete za logere.to mogu biti opasne greske ili neke male sitnice.

Sustina ovog napisanog teksta je da uzmemo source code ,kreiramo loger,instanciramo lekser I onda citamo znakove sa ulaza dok ne dodjemo do eof. Za svaki ispisemo vrednost I njegovu klasu.