**PROJEKAT PP1**

**Leksicka analiza**

**Src/sym.java**

Za sve leksicke structure pravimo konstantu kojoj dodeljujemo broj

public static final int LEKSICKE STRUKTURE = 1;

**spec/mjlexer.lex**

sastoji se iz 3 sekcije

* Import
* Direktiva
* Regularni izrazi

Generator na osnovu ove specifikacije genrise java implementaciju analizatora.

Zelimo da nasu impl.smestimo u nas paket – dopunjujemo import sekicju.

Nas analizator treba da bude CUP kompatibilan, treba da implementira cup skener interfejs.

%cup

Zelimo da brojimo linije i kolone:

%line

%column

Mozemo da u generisanu javu klasu ukljucimo nas usluzni kod, neke nase pomocne metode.

Pisemo u sekciji kod

%{

}%

I tu pisemo usluzne metode new\_symbol, koje su iz paketa java\_cup.runtime.Symbol

Nas lekser je kao automat, ima stanje u kom vrsi citanje ulaznih sekv i jedno za obradu komentara. Dodajemo %xstate

Kada lekser dodje do EOF moze da izvrsi neku akciju. Vratiti token EOF kad dodjemo do kraja:

%eofval{

return new\_symbol(sym.EOF);

%eofval}

Sekcija regularni izrazi:

Na osnovu njih omogucimo akcije kada se okine neki od regeksa

Zelimo da lekser ignorise sve vrste belih znakova u kodu

"\t" { }

"\r \n" { }

" " { }

"\b" { }

"\f" { }

Potom dodajemo ostale regularne izraze.

Za sve simbole je ok da bude <YYINITIAL> stanje, i ne mora da se pise.

Za komentar, kada naidje // treba da predjemo u COMMENT stanje, i da se vratimo u YYINITIAL tek kad naidjemo na \r\n

Na kraju svih regeksa, imamo hvatanje simbola koji nam ne predstavljaju nista – nedozvljene simbole:

. { System.err.println("Leksicka greska (" + yytext() + ")" u liniji + (yyline + 1)); }

U lib dodajemo cup.jar i jflex.jar, i dodajemo ih u java build path projekta.

New File -> build.xml -ant fajl koji ce nam generisati sta treba

Pisemo skriptu za bildovanje, doijamo Yylex.java

Imamo targete za delete, lexerGen i compile.

U test cemo dodati program.mj da testiramo.

Pisanje poruka u logove pomocu system.println cemo zameniti log4j Sa apache sajta

U src paket->dodajemo paket util za smestanje usluznih klasa za logovanje. Nju treba da konfigurisemo – config folder- u koji log fajl ispisujemo poruke i u kom format se ispisuje poruka.

Log4j.xml

**Sintaksna analiza**

Spec-> mjparser.cup. sastoji se iz 3 dela:

* Import
* Action core sekcija
* Sekcija gde pocinje gramatika sa deklarisanim terminalnim i neterminalnim simbolom.

Cilj da konstruisemo apstraktno sintaksno stablo

Neterminali – veliko pocetno slovo

Terminali – velika slova

::= odvaja levu i desnu stranu smene

Gramatika fajlova cup specifikacije sadrzi mogucnost da svakoj produkciji dodelimo ime koje ce se koristiti kao ime novogenerisane klase – na pocetku desne strane smene

Program ::= (Program) PROG IDENT VarDeclList LBRACE MethodDeclList RBRACE;

Deklarisati sve koriscene terminale i neterminale – iznad

terminal PROG, IDENT, LBRACE, RBRACE, ODENT, LPAREN, RPAREN;

nonterminal Program, VarDeclList, MerhodDeclList, VarDecl, Type, MethodDecl;

nonterminal FormPars, StatementList;

Leva rekurzija za varDeclList

VarDeclList ::= (VarDeclarations) VarDeclList VarDecl

|

(NoVarDeclarations) /\*epsilon\*/

;

VarDecl ::= (VarDecl) Type:varType IDENT:varName SEMI;

MerhodDeclList ::= (MethodDeclarations) MerhodDeclList MerhodDecl

|

(NoMethodDeclarations) /\*epsilon\*/

;

MerhodDecl ::= Type:retType IDENT:methName LPAREN FormPars RPAREN VarDeclList LBRACE StatementList RBRACE;

Sym.java ce biti izgenerisana automatski

Terminalne simbole nazivamo na isti nacin