|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 06** | **Syntax Analyzer (Top Down)** | | |
| Date of Exercise | \_\_\_\_\_\_\_\_\_\_\_ | Date of Output Verification | \_\_\_\_\_\_\_\_\_\_\_ |

**Question**

Write a program to construct predictive parsing table for the given grammar.

**Program**

import java.io.\*;

import java.util.\*;

//@author William Scott

public class CompilerLabExp6SyntaxAnalyzerTopDown {

//UR12CS135 - P.William Scott - Exp 6 - Syntax Analyzer

public static BufferedReader brip, brip2;

public static String ts = "", tsa[];

public static int ti = 0;

public static char ct, cc;

public static ArrayList ipfrom = new ArrayList(), ipto = new ArrayList(), states = new ArrayList(), symbols = new ArrayList(), first = new ArrayList(), follow = new ArrayList();

public static void main(String[] args) throws Exception {

System.out.println("\nUR12CS135 - Syntax Analyzer (Top Down)");

readip();

inititalize();

findfirst();

findfollow();

showfirstfollow();

constructtable();

}

public static void constructtable() {

System.out.println("\n-------Table-------");

getsymbols();

String[][] a = new String[states.size() + 1][symbols.size() + 1];

for (int i = 0; i < states.size() + 1; i++) {

for (int j = 0; j < symbols.size() + 1; j++) {

a[i][j] = "";

}

}

System.out.print(" ");

for (Object symbol : symbols) {

System.out.print("\t" + symbol);

}

System.out.println("");

for (int p = 0; p < states.size(); p++) {

char t;

String h;

System.out.print(states.get(p));

for (int i = 0; i < ipfrom.size(); i++) {

if (ipfrom.get(i).toString().contains(states.get(p).toString())) {

t = ipto.get(i).toString().charAt(0);

if (states.contains(t + "")) {

int q = states.indexOf(t + "");

h = first.get(q).toString();

} else {

h = t + "";

}

for (int j = 0; j < h.length(); j++) {

for (int k = 0; k < symbols.size(); k++) {

if (h.charAt(j) == symbols.get(k).toString().charAt(0)) {

a[p][k] += "" + (i + 1) + "";

}

}

}

}

}

System.out.print("\t");

for (int i = 0; i < symbols.size() - 1; i++) {

if (a[p][i].length() < 1) {

System.out.print("\t");

} else {

System.out.print(a[p][i] + "\t");

}

}

System.out.println("");

}

}

public static void getsymbols() {

for (Object ipto1 : ipto) {

String p = ipto1.toString();

for (int i = 0; i < p.length(); i++) {

if (!states.contains(p.charAt(i) + "")) {

symbols.add(p.charAt(i));

}

}

}

symbols.add("$");

}

public static void readip() throws Exception {

System.out.println("\n---------Input File---------");

int y = 1;

brip = new BufferedReader(new FileReader("..\\Exp 6 - Predictive Parsing.txt"));

while ((ts = brip.readLine()) != null) {

System.out.println(y + ") " + ts);

tsa = ts.split(" ");

ipfrom.add(tsa[0]);

ipto.add(tsa[1]);

y++;

}

}

public static void inititalize() {

for (int i = 0; i < 20; i++) {

first.add("");

follow.add("");

}

}

public static void findfirst() {

for (Object fo : ipfrom) {

if (states.contains(fo)) {

ti = states.indexOf(fo);

} else {

states.add(fo);

ti = states.size() - 1;

}

addfirst(fo.toString(), 0, "");

}

}

public static void findfollow() {

for (Object state : states) {

cc = state.toString().charAt(0);

if (states.get(0).toString().contains(cc + "")) {

addfollowsym('$');

}

addfollow();

}

}

public static void addfollow() {

for (int i = 0; i < ipto.size(); i++) {

ts = ipto.get(i).toString();

if (ts.contains(cc + "")) {

unnamed(' ', i); //Unfinished nor will i finish it

}

}

}

public static void unnamed(char a, int i) {

char p;

ti = ts.lastIndexOf(cc);

ti = ts.indexOf(cc);

if (ti == ts.length() - 1) {

getvaluesforfollow(ipfrom.get(i).toString().charAt(0), "follow");

} else if (!states.contains(ts.charAt(ti + 1) + "")) {

addfollowsym(ts.charAt(ti + 1));

} else {

addfirsttofollow(ts.charAt(ti + 1));

}

}

public static void addfirsttofollow(char a) {

int y = states.indexOf(a + "");

for (int i = 0; i < first.get(y).toString().length(); i++) {

addfollowsym(first.get(y).toString().charAt(i));

}

}

public static void getvaluesforfollow(char a, String f) {

int t, r;

String p;

t = states.indexOf(cc + "");

r = states.indexOf(a + "");

if (f.equals("follow")) {

p = follow.get(r).toString();

} else {

p = first.get(r).toString();

}

for (char c : p.toCharArray()) {

addfollowsym(c);

}

}

public static void addfollowsym(char a) {

int t;

String g;

t = states.indexOf(cc + "");

if (a == '~') {

} else if (!follow.get(t).toString().contains(a + "")) {

g = follow.get(t).toString() + a;

follow.set(t, g);

}

}

public static void addfirst(String a, int b, Object c) {

char h = ' ';

for (int i = b; i < ipfrom.size(); i++) {

if (ipfrom.get(i).toString().contains(a)) {

ct = ipto.get(i).toString().charAt(0);

if (ipto.get(i).toString().length() > 1) {

h = ipto.get(i).toString().charAt(1);

}

if (ipfrom.contains(ct + "")) {

addfirst(ct + "", ++i, h);

} else if (!first.get(ti).toString().contains(ct + "")) {

ts = first.get(ti).toString() + ct;

first.set(ti, ts);

}

}

}

}

public static void showfirstfollow() {

System.out.println("\n----------First & Follow---------");

System.out.println("State\tFirst\tFollow");

for (int i = 0; i < states.size(); i++) {

System.out.println(" " + states.get(i) + "\t " + first.get(i) + "\t " + follow.get(i));

}

}

}

**Input**

E +T

E T

T @F

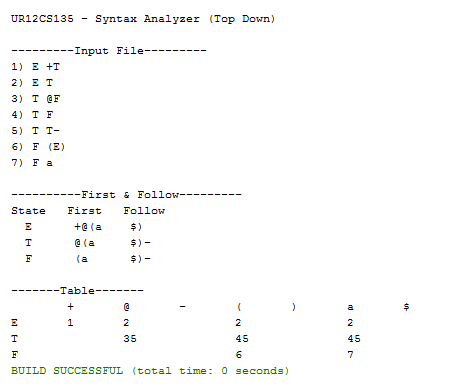
T F

T T-

F (E)

F a

**Output**



**Result**

Implementation of syntax analyzer is successfully done.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Jeban Chandir Moses

Date: \*\* – \*\* – 2015