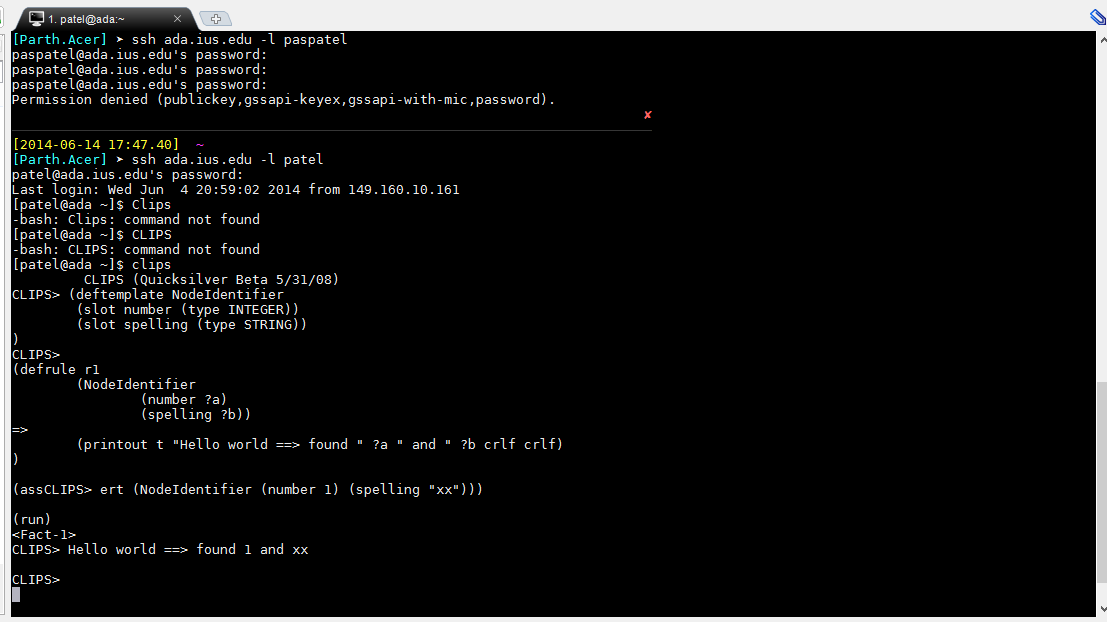
**Part 1 Clips Program Documentation.**



**Part 2 and Part 3**

import java.util.Arrays;

import java.util.ArrayList;

import java.util.Scanner;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Parth

\*/

public class P4 {

static String token\_while = "while";

static String token = "-1";

static String token\_semicolon = ";";

static String token\_id = "";

static String token\_else = "else";

static String token\_assignop = ":=";

static String token\_if = "if";

static String token\_do = "do";

static String token\_begin = "begin";

static String token\_end = "end";

static String token\_then = "then";

static int count = 0;

static int gCount = 1;

static ArrayList<String> tokens = new ArrayList<String>();

static String[] keyTokens = {token\_semicolon, token\_assignop, token\_while, token\_do, token\_begin, token\_end, token\_if, token\_then, token\_else};

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

token = getToken();

Parse\_P();

}

public static void accept(String n, String message){

if (n.equals(token))

System.out.println("Accepted: " + n);

else

System.out.println("error\*\* " + message);

}

public static void Parse\_P(){

Parse\_L();

}

public static void Parse\_L(){

Parse\_S();

while(token.equals(token\_semicolon)){

accept(token\_semicolon, "\*\*101\*\*)");

Parse\_S();

}

}

public static void Parse\_S(){

if (token.equals(token\_id)){

accept(token\_id, "\*\*102\*\*");

tokens.add(token);

token = getToken();

accept(token\_assignop, "\*\*103 expected :=\*\*");

tokens.add(token);

token = getToken();

accept(token\_id,"\*\*104\*\*");

tokens.add(token);

intRecord(tokens);

generateFile(tokens);

}

else if(token.equals(token\_while)){

accept(token\_while,"\*\*105\*\*");

token = getToken();

accept(token\_id,"\*\*106 ID expected\*\*");

token = getToken();

accept(token\_do,"\*\*107 Do expected\*\*");

Parse\_S();

}

else if(token.equals(token\_begin)){

accept(token\_begin,"\*\*115\*\*");

token = getToken();

Parse\_L();

token = getToken();

accept(token\_end,"\*\*116 End expected\*\*");

}

else if(token.equals(token\_if)){

accept(token\_if,"\*\*108\*\*");

token = getToken();

accept(token\_id,"\*\*109 ID expected\*\*");

token = getToken();

accept(token\_then, "\*\*110 Then expected\*\*");

token = getToken();

Parse\_S();

}

else if(token.equals(token\_if)){

accept(token\_if,"\*\*111\*\*");

token = getToken();

accept(token\_id,"\*\*112 Id expected\*\*");

token = getToken();

accept(token\_then,"\*\*113 Then expected\*\*");

token = getToken();

Parse\_S();

token = getToken();

accept(token\_else,"\*\*114 Else\*\*");

token = getToken();

Parse\_S();

}

}

public static String getToken(){

File file = new File("file.txt");

String temp = "";

try{

Scanner input = new Scanner(file);

if(count > 0){

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

input.next();

}

}

token = input.next();

if(token.equals(";")){

count = count + 1;

return token\_semicolon;

}else if(token.equals(":=")){

count = count + 1;

return token\_assignop;

}else if(token.equals("while")){

count = count + 1;

return token\_while;

}else if(token.equals("do")){

count = count + 1;

return token\_do;

}else if (token.equals("begin")){

count = count + 1;

return token\_begin;

}else if (token.equals("end")){

count = count + 1;

return token\_end;

}else if (token.equals("if")){

count = count + 1;

return token\_if;

}else if(token.equals("then")){

return token\_then;

}else if(token.equals("else")){

count = count + 1;

return token\_else;

}else{

count = count + 1;

token\_id = token;

return token\_id;

}

}

catch(Exception e){

e.printStackTrace();

}

System.out.println("Token: " + token);

System.out.println("Leaving getToken");

count = count + 1;

return token;

}

public static void intRecord(ArrayList<String> sTok){

System.out.println("(deftemplate NodeIdentifier ");

System.out.println("\t(slot number (type INTEGER))");

System.out.println("\t(slot spelling (type STRING))");

System.out.println(")");

System.out.println("(defrule r1");

System.out.println("\t(NodeIdentifier");

System.out.println("\t\t(number ?a)");

System.out.println("\t\t(spelling ?b))");

System.out.println("=>");

System.out.println("\t(printout \"Hello world ==> found \" ?a \" and \" ?b)");

System.out.println(")");

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

System.out.println("(assert (NodeIdentifier (number "+i+") (spelling \""+sTok.get(i)+"\")))");

}

}

private static void generateFile(ArrayList<String> tokens2) {

try {

File file = new File("output.txt");

if(!file.exists()){

file.createNewFile();

}

FileWriter out = new FileWriter(file.getAbsoluteFile());

BufferedWriter bw = new BufferedWriter(out);

bw.write("(deftemplate NodeIdentifier \n");

bw.write("\t(slot number (type INTEGER))\n");

bw.write("\t(slot spelling (type STRING))\n");

bw.write(")\n");

bw.write("(defrule r1\n");

bw.write("\t(NodeIdentifier\n");

bw.write("\t\t(number ?a)\n");

bw.write("\t\t(spelling ?b))\n");

bw.write("=>\n");

bw.write("\t(printout \"Hello world ==> found \" ?a \" and \" ?b)\n");

bw.write(")\n");

for(int i=0; i<tokens2.size(); i++){

bw.write("(assert (NodeIdentifier (number "+i+") (spelling \""+tokens2.get(i)+"\")))\n");

}

bw.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

**Input: -**

while begin found if then ;

do = now

**Output: -**

(deftemplate NodeIdentifier

(slot number (type INTEGER))

(slot spelling (type STRING))

)

(defrule r1

(NodeIdentifier

(number ?a)

(spelling ?b))

=>

(printout "Hello world ==> found " ?a " and " ?b)

)

(assert (NodeIdentifier (number 0) (spelling "found")))

(assert (NodeIdentifier (number 1) (spelling "if")))

(assert (NodeIdentifier (number 2) (spelling "then")))

Saves the output to a file too ..

