**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

PRÁCTICA 2

Máquina de pila

**Profesor:**

• **Tecla Parra Roberto**

**Alumno:**

* **Pacheco Delgado José Jacobeth**

**Grupo:**

• **3CM8**

**Materia:**

• **Compiladores**

**Introducción**

La práctica 2 consiste en el uso de java para modo gráfico, se cuenta con una serie de librerías en java. Graficadora de figuras básicas (línea recta, circulo y rectángulo).

**Objetivos**

* Usar la carpeta grafibasi para dibujar círculos, líneas, y rectángulos usando polimorfismo
* Modificar los archivos **Maquina.java** y el archivo de la gramática (**forma.y**) para obtener el resultado requerido.

**Desarrollo**

Primero se tuvo que hacer la modificación en la especificación de yacc para aceptar los símbolos terminales después del nombre de una figura, se agregaron 3 terminales a la figura circulo, para aceptar el radio, posición en x y posición en y. En la figura línea, se agregaron cuatro terminales, dos para las coordenadas del punto inicial de la recta y otros dos para las coordenadas del punto final. En el caso del rectángulo se agregaron otros cuatro, uno para la posición en x, otro en y y finalmente para el alto y el ancho de la figura.

El código para correr en mi caso es (Por el sistema operativo)

./byacc –J forma.y

Java Maquina.java

(Eliminar el break de la línea 357) y listo

forma.y

%{

**import** **java.lang.Math**;

**import** **java.io.**\*;

**import** **java.util.StringTokenizer**;

**import** **java.awt.**\*;

**import** **java.awt.event.**\*;

**import** **javax.imageio.**\*;

**import** **java.awt.image.**\*;

**import** **javax.swing.**\*;

%}

%token NUMBER LINE CIRCULO RECTANGULO COLOR PRINT IMAGEN FILTRO

%start list

%%

list :

| list ';'

| list inst ';' {

maq.code("print"); maq.code("STOP"); **return** 1 ;

}

;

inst: NUMBER { ((Algo)$$.obj).inst=maq.code("constpush");

maq.code(((Algo)$1.obj).simb); }

| RECTANGULO NUMBER NUMBER NUMBER NUMBER {

maq.code("constpush");

maq.code(((Algo)$2.obj).simb);

maq.code("constpush");

maq.code(((Algo)$3.obj).simb);

maq.code("constpush");

maq.code(((Algo)$4.obj).simb);

maq.code("constpush");

maq.code(((Algo)$5.obj).simb);

maq.code("rectangulo");

}

| LINE NUMBER NUMBER NUMBER NUMBER NUMBER{

maq.code("constpush");

maq.code(((Algo)$2.obj).simb);

maq.code("constpush");

maq.code(((Algo)$3.obj).simb);

maq.code("constpush");

maq.code(((Algo)$4.obj).simb);

maq.code("constpush");

maq.code(((Algo)$5.obj).simb);

maq.code("constpush");

maq.code(((Algo)$6.obj).simb);

maq.code("line");

}

| CIRCULO NUMBER NUMBER NUMBER{

maq.code("constpush");

maq.code(((Algo)$2.obj).simb);

maq.code("constpush");

maq.code(((Algo)$3.obj).simb);

maq.code("constpush");

maq.code(((Algo)$4.obj).simb);

maq.code("circulo");

}

| COLOR NUMBER { maq.code("constpush");

maq.code(((Algo)$2.obj).simb); maq.code("color");}

| IMAGEN NUMBER NUMBER {

Simbolo s=new Simbolo();

s.ponDibu(new Imagen(leeImagen("dalmata.jpg"),

(int)((Algo)$2.obj).simb.val,

(int)((Algo)$3.obj).simb.val, jf));

maq.code("varpush");

maq.code(s);

maq.code("draw");

}

| FILTRO NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER

NUMBER NUMBER NUMBER {

maq.code("constpush");

maq.code(((Algo)$2.obj).simb);

maq.code("constpush");

maq.code(((Algo)$3.obj).simb);

maq.code("constpush");

maq.code(((Algo)$4.obj).simb);

maq.code("constpush");

maq.code(((Algo)$5.obj).simb);

maq.code("constpush");

maq.code(((Algo)$6.obj).simb);

maq.code("constpush");

maq.code(((Algo)$7.obj).simb);

maq.code("constpush");

maq.code(((Algo)$8.obj).simb);

maq.code("constpush");

maq.code(((Algo)$9.obj).simb);

maq.code("constpush");

maq.code(((Algo)$10.obj).simb);

Simbolo s=new Simbolo();

s.ponDibu(new Imagen(leeImagen("dalmata.jpg"),

100,

100, jf));

maq.code("varpush");

maq.code(s);

maq.code("filtro");

}

;

%%

public BufferedImage leeImagen(String nombre){

BufferedImage imagen;

**try** {

imagen = ImageIO.read(new File(nombre));

} catch (IOException e){

System.err.println("no cargo "+nombre);

**return** null;

}

**return** imagen;

}

**class** **Algo** {

Simbolo simb;

int inst;

Dibujable dibu;

public Algo(int i){ inst=i; }

public Algo(Dibujable d){ dibu=d; }

public Algo(Simbolo s, int i){

simb=s; inst=i;

}

}

public void setTokenizer(StringTokenizer st){

this.st= st;

}

public void setNewline(boolean newline){

this.newline= newline;

}

Tabla tabla;

Maquina maq;

StringTokenizer st;

boolean newline;

int yylex(){

String s;

int tok;

Double d;

Simbolo simbo;

**if** (!st.hasMoreTokens())

**if** (!newline) {

newline=true;

**return** ';';

}

**else**

**return** 0;

s = st.nextToken();

**try** {

d = Double.valueOf(s);

yylval = new ParserVal(

new Algo(tabla.install("", NUMBER, d.doubleValue()),0) );

tok = NUMBER;

} catch (**Exception** e){

**if**(Character.isLetter(s.charAt(0))){

System.out.println("s es = ("+s+")");

**if**((simbo=tabla.lookup(s))==null)

yylval = new ParserVal(new Algo(simbo, 0));

tok= simbo.tipo;

} **else** {

tok = s.charAt(0);

}

}

**return** tok;

}

void yyerror(String s){

System.out.println("parser error: "+s);

}

static Parser par = new Parser(0);

static JFrame jf;

static JLabel lmuestra=new JLabel(" ");

static Canvas canv;

static Graphics g;

Parser(int foo){

maq=new Maquina();

tabla=new Tabla();

tabla.install("line", LINE, 0.0);

tabla.install("circulo", CIRCULO, 0.0);

tabla.install("rectangulo", RECTANGULO, 0.0);

tabla.install("color", COLOR, 0.0);

tabla.install("print", PRINT, 0.0);

tabla.install("ima", IMAGEN, 0.0);

tabla.install("fil", FILTRO, 0.0);

maq.setTabla(tabla);

jf=new JFrame("Calcula");

canv=new Canvas();

canv.setSize(600,600);

jf.add("North", new PanelEjecuta(maq, this));

jf.add("Center", canv);

jf.setSize( 600, 700);

jf.setVisible(true);

g=canv.getGraphics();

maq.setGraphics(g);

jf.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

}

public static void main(String args[]){ new Parser(); }

Como se observa, se agrega como símbolo terminal al elemento racionalnum y operaciones (**op**) que será el operador para realizar las operaciones especificadas.

De igual forma en el caso de expresión se modificaron las acciones gramaticales, donde en caso de obtener un número racional este se imprimirá en pantalla para facilitar el observar los resultados de las operaciones. En el caso de las operaciones, las acciones gramaticales mandan llamar a los métodos especificados en **racional\_cal.c**, dichos métodos llevan a cabo las operaciones entre los numeradores y denominadores de los números racionales a operar.

%option noyywrap

%{

#include <stdio.h>

#include <stdlib.h>

#include "racional\_cal.h"

#include "y.tab.h"

void RmWs(char\* str)*;*

extern YYSTYPE yylval*;*

%}

/\* Add your Flex definitions here \*/

/\* Some definitions are already provided to you\*/

op [-+\*/()]

ws [ \t]+

digits [0-9]

**number** (0|[1-9]+{digits}\*)

sub [n]

fr [+]

racionalnum {ws}\*{sub}?{ws}\*{number}{ws}\*{fr}{ws}\*{number}{ws}\*

%%

{racionalnum} {

int num, den;

RmWs(yytext);

printf("(%s)", yytext);

sscanf(yytext,"%d %d", &num, &den);

yylval=creaRacional(num, den);

return racionalnum;}

{op} |

\n {return \*yytext*;}*

{ws} { /\* Do nothing \*/ }

. { /\* Do nothing \*/ }

%%

void RmWs(char\* str) {

int i = 0, j = 0*;*

char temp[strlen(str) + 1]*;*

strcpy(temp, str)*;*

while (temp[i] != '\0') {

while (temp[i] == ' ')

i++*;*

str[j] = temp[i]*;*

i++*;*

j++*;*

}

str[j] = '\0'*;*

}

Como se observa, se agregaron las constantes obtenidas de las acciones gramaticales, cada una de ellas se va sacando de la pila para posteriormente pasar a la función de graficar dichos parámetros para realizar el dimensionamiento de las figuras y el desplazamient

Maquina.java

#include <stdio.h>

#include "racional\_cal.h"

#include <stdlib.h>

int main() { **return** yyparse(); }

int yyerror(**const** char\* s) {

printf("%s**\n**", s);

**return** 0;

}

Racional \*creaRacional(int num, int den){

Racional \*nvo;

**if**(den==0)

**return** (Racional \*)NULL;

nvo=(Racional \*)malloc(**sizeof**(Racional));

**if**(!nvo){

puts("no hay memoria para crear Racional ");

**return** (Racional \*)NULL;

}

nvo->num=num;

nvo->den=den;

**return** nvo;

}

void asignar(Racional \*r, int num, int den){

r -> num = num; r -> den = den;

}

int numerador(Racional \*r){ **return** r ->num; }

int denominador(Racional \*r){ **return** r ->den; }

Racional\* racionalSuma(Racional \*r, Racional \*s){

int nvonum = (r -> num \* s -> den) + (s -> num \* r -> den);

int nvoden = r -> den \* s -> den;

Racional \*nvo = creaRacional(nvonum, nvoden);

**return** nvo;

}

Racional\* racionalResta(Racional \*r, Racional \*s){

int nvonum = (r -> num \* s -> den) - (s -> num \* r -> den);

int nvoden = r -> den \* s -> den;

Racional \*nvo = creaRacional(nvonum, nvoden);

**return** nvo;

}

Racional\* racionalMultiplicar(Racional \*r, Racional \*s){

int nvonum = r -> num \* s -> num;

int nvoden = r -> den \* s -> den;

Racional \*nvo = creaRacional(nvonum, nvoden);

**return** nvo;

}

Racional\* racionalDividir(Racional \*r, Racional \*s){

int nvonum = r -> num \* s -> den;

int nvoden = r -> den \* s -> num;

Racional \*nvo = creaRacional(nvonum, nvoden);

**return** nvo;

}

int esIgual(Racional \*r, Racional \*s){

**return** (r -> num \* s -> den) == (r -> den \* s -> num);

}

void imprimirR(void \*r){

Racional \*p = (Racional\*)r;

printf(" = (%d / %d)**\n**",p -> num, p -> den);

}

Racional \*copiar(Racional \*r){

**return** creaRacional(r -> num, r -> den);

}

Adémas para agilizar las pruebas fue creado un archivo **Makefile** ( es un archivo (necesariamente llamado **Makefile**) que contiene una serie de **directivas** e **instrucciones** utilizadas por una **automatización** de acumulación, herramienta para generar **un objetivo / meta**)

Gram=y.tab.c y.tab.h

all: $(Gram) lex.yy.c racional\_cal.c

@gcc -o racio y.tab.c lex.yy.c racional\_cal.c

@echo Compiled

racional\_cal.c:

@echo "Si esta"

**$(**Gram**)**: racional\_cal.y

@yacc -d racional\_cal.y

lex.yy.c: racional\_cal.l

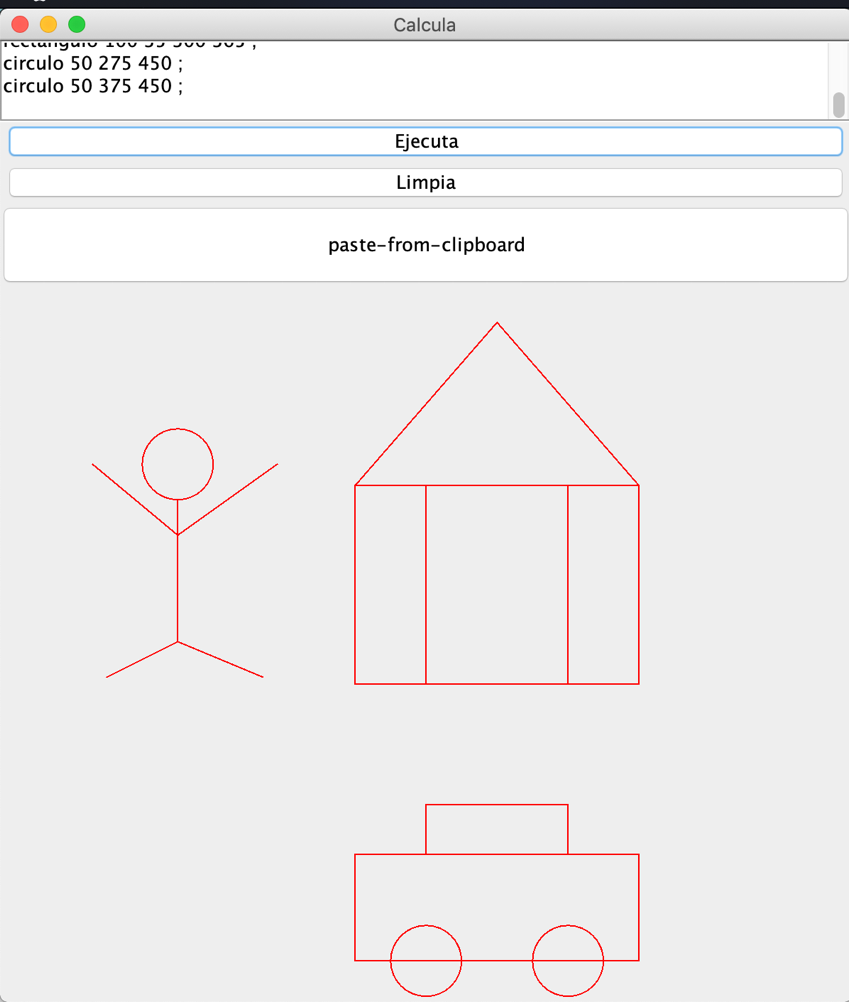
@flex racional\_cal.l

clean:

@rm -f \*.out lex.yy.c \*.tab.\* racio

@echo Clean

Exitosamente fueron añadidos y modificado los elementos para dimensionar y desplazar a las figuras básicas. De lado izquierdo se muestran los elementos con los que se realizó este pequeño dibujo.



**Conclusión**

El entorno de java en muy distinto al de c, pero nos ofrece facilidades gráficas mediante clases usadas con polimorfismo, adémas de contar con su propio yacc en este caso llamado byacc.