**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Compiladores.**

**Tema:**

**Práctica 2.**

**(Reporte).**

**Profesor:**

**Roberto Tecla Parra.**

**Alumno:**

**Mario Alberto Miranda Sandoval.**

**Grupo:**

**3CM7.**

**Fecha:**

**03 – Febrero – 2020.**

**Objetivo de la práctica.**

**Dibujando con Java**

**Descripción**

Para esta práctica nuevamente hemos hecho uso de una aplicación en YACC para generar un programa que nos permita dibujar figuras básicas como:

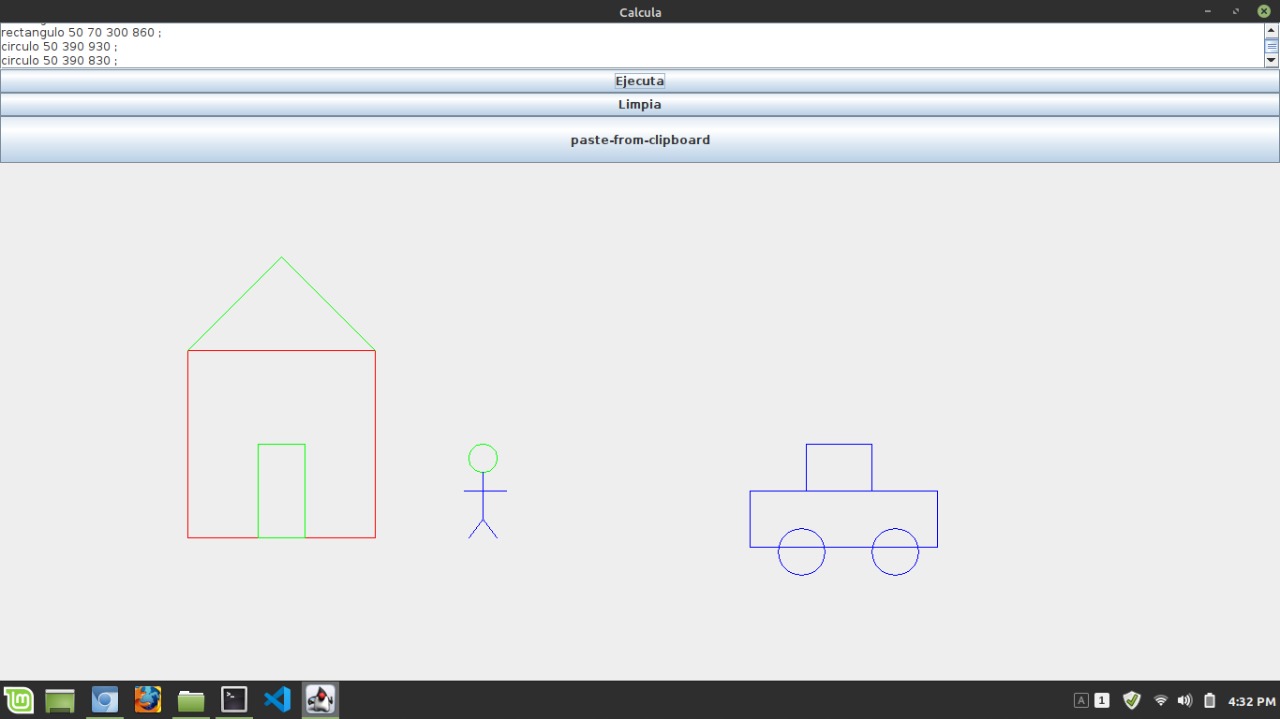
1. Círculos

2. Rectángulos

3. Líneas

**Ejemplos**

A continuación, muestro una captura de pantalla, la cual muestra la compilación del código en YACC, y también la compilación del código que es generado en java y finalmente la ejecución del programa.

****

**Modificaciones al código.**

1. | RECTANGULO NUMBER NUMBER NUMBER NUMBER  {
2. maq.code("constpush");
3. maq.code(((Algo)$2.obj).simb);
5. maq.code("constpush");
6. maq.code(((Algo)$3.obj).simb);
8. maq.code("constpush");
9. maq.code(((Algo)$4.obj).simb);
11. maq.code("constpush");
12. maq.code(((Algo)$5.obj).simb);
14. maq.code("rectangulo");
15. }

Como se puede ver, se modificó el rectángulo haciendo que fuese capaz de recibir más tokens, también en las acciones gramaticales se añadió más push a la pila con los tokens correspondientes.

1. **void** rectangulo(){
2. **double** X, Y, ancho, alto;
3. //Obtenemos el valor de la posición en X haciendo pop de la pila
4. X = ((Double)pila.pop()).doubleValue();
5. //Obtenemos el valor de la posición en Y haciendo pop de la pila
6. Y = ((Double)pila.pop()).doubleValue();
7. //Obtenemos el valor de la anchura del rectangulo haciendo pop de la pila
8. ancho = ((Double)pila.pop()).doubleValue();
9. //Obtenemos el valor de la altura dle rectangulo haciendo pop de la pila
10. alto = ((Double)pila.pop()).doubleValue();
11. **if**(g!=null){
12. (   **new** Rectangulo((**int**)X, (**int**)Y, (**int**)ancho, (**int**)alto )  ).dibuja(g);
13. }
14. }

Ahora en la máquina, se modifico el desempilaminto de los valores para después mandarlos a dibujar con su correspondiente instrucción en java.

Lo mismo se hizo para la línea y el círculo.

**Código.**

**Forma.y**

1. %{
2. **import** java.lang.Math;
3. **import** java.io.\*;
4. **import** java.util.StringTokenizer;
5. **import** java.awt.\*;
6. **import** java.awt.event.\*;
7. **import** javax.swing.\*;
8. %}
9. %token NUMBER LINE CIRCULO RECTANGULO COLOR PRINT
10. %start list
11. %%
12. list :
13. | list ';'
14. | list inst ';'   {
15. maq.code("print");
16. maq.code("STOP");
17. **return** 1 ;
18. }
19. ;
20. inst:  NUMBER  { ((Algo)$$.obj).inst=maq.code("constpush");
21. maq.code(((Algo)$1.obj).simb); }

24. | RECTANGULO NUMBER NUMBER NUMBER NUMBER  {
25. maq.code("constpush");
26. maq.code(((Algo)$2.obj).simb);
28. maq.code("constpush");
29. maq.code(((Algo)$3.obj).simb);
31. maq.code("constpush");
32. maq.code(((Algo)$4.obj).simb);
34. maq.code("constpush");
35. maq.code(((Algo)$5.obj).simb);
37. maq.code("rectangulo");
38. }
40. | LINE NUMBER NUMBER NUMBER NUMBER {
41. maq.code("constpush");
42. maq.code(((Algo)$2.obj).simb);
44. maq.code("constpush");
45. maq.code(((Algo)$2.obj).simb);
47. maq.code("constpush");
48. maq.code(((Algo)$3.obj).simb);
50. maq.code("constpush");
51. maq.code(((Algo)$4.obj).simb);
53. maq.code("constpush");
54. maq.code(((Algo)$5.obj).simb);
56. maq.code("line");
57. }
59. | CIRCULO NUMBER NUMBER NUMBER {
60. maq.code("constpush");
61. maq.code(((Algo)$2.obj).simb);
63. maq.code("constpush");
64. maq.code(((Algo)$3.obj).simb);
66. maq.code("constpush");
67. maq.code(((Algo)$4.obj).simb);
69. maq.code("circulo");}
71. | COLOR NUMBER { maq.code("constpush");
72. maq.code(((Algo)$2.obj).simb); maq.code("color");}
73. ;
74. %%
75. **class** Algo {
76. Simbolo simb;
77. **int** inst;
78. **public** Algo(**int** i){ inst=i; }
79. **public** Algo(Simbolo s, **int** i){
80. simb=s; inst=i;
81. }
82. }
83. **public** **void** setTokenizer(StringTokenizer st){
84. **this**.st= st;
85. }
86. **public** **void** setNewline(**boolean** newline){
87. **this**.newline= newline;
88. }
89. Tabla tabla;
90. Maquina maq;
92. StringTokenizer st;
93. **boolean** newline;
94. **int** yylex(){
95. String s;
96. **int** tok;
97. Double d;
98. Simbolo simbo;
99. **if** (!st.hasMoreTokens())
100. **if** (!newline) {
101. newline=**true**;
102. **return** ';';
103. }
104. **else**
105. **return** 0;
106. s = st.nextToken();
107. **try** {
108. d = Double.valueOf(s);
109. yylval = **new** ParserVal(
110. **new** Algo(tabla.install("", NUMBER, d.doubleValue()),0) );
111. tok = NUMBER;
112. } **catch** (Exception e){
113. **if**(Character.isLetter(s.charAt(0))){
114. **if**((simbo=tabla.lookup(s))==**null**)
115. yylval = **new** ParserVal(**new** Algo(simbo, 0));
116. tok= simbo.tipo;
117. } **else** {
118. tok = s.charAt(0);
119. }
120. }
121. **return** tok;
122. }
123. **void** yyerror(String s){
124. System.out.println("parser error: "+s);
125. }
126. **static** Parser par = **new** Parser(0);
127. **static** JFrame jf;
128. **static** JLabel lmuestra=**new** JLabel("                                 ");
129. **static** Canvas canv;
130. **static** Graphics g;
131. Parser(**int** foo){
132. maq=**new** Maquina();
133. tabla=**new** Tabla();
134. tabla.install("line", LINE, 0.0);
135. tabla.install("circulo", CIRCULO, 0.0);
136. tabla.install("rectangulo", RECTANGULO, 0.0);
137. tabla.install("color", COLOR, 0.0);
138. tabla.install("print", PRINT, 0.0);
139. maq.setTabla(tabla);
140. jf=**new** JFrame("Calcula");
141. canv=**new** Canvas();
142. canv.setSize(600,600);
143. jf.add("North", **new** PanelEjecuta(maq, **this**));
144. jf.add("Center", canv);
145. jf.setSize( 600, 700);
146. jf.setVisible(**true**);
147. g=canv.getGraphics();
148. maq.setGraphics(g);
149. jf.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);
150. }
151. **public** **static** **void** main(String args[]){ **new** Parser(); }

**Maquina.java**

1. **import** java.awt.\*;
2. **import** java.util.\*;
3. **import** java.lang.reflect.\*;
5. **class**  Maquina {
6. Tabla tabla;
7. Stack pila;
8. Vector prog;
10. **static** **int** pc=0;
11. **int** progbase=0;
12. **boolean** returning=**false**;
14. Method metodo;
15. Method metodos[];
16. Class c;
17. Graphics g;
18. **double** angulo;
19. **int** x=0, y=0;
20. Class parames[];
22. Maquina(){
23. }
25. **public** **void** setTabla(Tabla t){
26. tabla = t;
27. }
29. **public** **void** setGraphics(Graphics g){
30. **this**.g=g;
31. }
33. Maquina(Graphics g){
34. **this**.g=g;
35. }
37. **public** Vector getProg(){
38. **return** prog;
39. }
41. **void** initcode(){
42. pila=**new** Stack();
43. prog=**new** Vector();
44. }
46. Object pop(){
47. **return** pila.pop();
48. }
50. **int** code(Object f){
51. System.out.println("Gen ("+f+") size="+prog.size());
52. prog.addElement(f);
53. **return** prog.size()-1;
54. }
56. **void** execute(**int** p){
57. String inst;
58. System.out.println("progsize="+prog.size());
59. **for**(pc=0;pc < prog.size(); pc=pc+1){
60. System.out.println("pc="+pc+" inst "+prog.elementAt(pc));
61. }
62. **for**(pc=p; !(inst=(String)prog.elementAt(pc)).equals("STOP") && !returning;){
63. //for(pc=p;pc < prog.size();){
64. **try** {
65. //System.out.println("111 pc= "+pc);
66. inst=(String)prog.elementAt(pc);
67. pc=pc+1;
68. System.out.println("222 pc= "+pc+" instr "+inst);
69. c=**this**.getClass();
70. //System.out.println("clase "+c.getName());
71. metodo=c.getDeclaredMethod(inst, **null**);
72. metodo.invoke(**this**, **null**);
73. }
74. **catch**(NoSuchMethodException e){
75. System.out.println("No metodo "+e);
76. }
77. **catch**(InvocationTargetException e){
78. System.out.println(e);
79. }
80. **catch**(IllegalAccessException e){
81. System.out.println(e);
82. }
83. }
84. }
86. **void** constpush(){
87. Simbolo s;
88. Double d;
89. s=(Simbolo)prog.elementAt(pc);
90. pc=pc+1;
91. pila.push(**new** Double(s.val));
92. }
94. **void** color(){
95. Color colors[]={Color.red,Color.green,Color.blue};
96. **double** d1;
97. d1=((Double)pila.pop()).doubleValue();
98. **if**(g!=**null**){
99. g.setColor(colors[(**int**)d1]);
100. }
101. }
103. /\*\*
104. \* Para nuestro caso una línea está compuesta por:
105. \* X1, Y1, X2, Y2
106. \*\*/
107. **void** line(){
108. **double** X1, Y1, X2, Y2;
109. //Obtenemos el primer valor, haciendo pop de la pila
110. X1 = ((Double)pila.pop()).doubleValue();
111. //Obtenemos el segundo valor, haciendo pop de la pila
112. Y1 = ((Double)pila.pop()).doubleValue();
113. //Obtenemos el tercer valor, haciendo pop de la pila
114. X2 = ((Double)pila.pop()).doubleValue();
115. //Obtenemos el cuarto valor, haciendo pop de la pila
116. Y2 = ((Double)pila.pop()).doubleValue();
118. //Los gráficos no deben ser nulos para poder dibujar
119. **if**(g!=**null**){
120. //Creamos un objeto Linea con los datos obtenidos de la pila
121. (   **new** Linea((**int**)X1, (**int**)Y1, (**int**)X2, (**int**)Y2)   ).dibuja(g);
122. }
123. }
125. /\*\*
126. \* Para nuestro caso un circulo  está compuesto por:
127. \* radio, X, Y
128. \*\*/
129. **void** circulo(){
130. **double** radio, X, Y;
131. //Obtenemos el valor del radio haciendo pop de la pila
132. radio = ((Double)pila.pop()).doubleValue();
133. //Obtenemos el valor de la posición  X haciendo pop de la pila
134. X = ((Double)pila.pop()).doubleValue();
135. //Obtenemos el valor de la posición Y haciendo pop de la pila
136. Y = ((Double)pila.pop()).doubleValue();
137. //Para poder dibujar la variable g no debe ser nula
138. **if**(g!=**null**){
139. //Creamos un nuevo objeto circulo
140. (   **new** Circulo((**int**)radio, (**int**)X, (**int**)Y)    ).dibuja(g);
141. }
142. }
144. /\*\*
145. \* Para nuestro caso un rectangulo está compuesto por:
146. \* X, Y, ancho, alto
147. \*\*/
148. **void** rectangulo(){
149. **double** X, Y, ancho, alto;
150. //Obtenemos el valor de la posición en X haciendo pop de la pila
151. X = ((Double)pila.pop()).doubleValue();
152. //Obtenemos el valor de la posición en Y haciendo pop de la pila
153. Y = ((Double)pila.pop()).doubleValue();
154. //Obtenemos el valor de la anchura del rectangulo haciendo pop de la pila
155. ancho = ((Double)pila.pop()).doubleValue();
156. //Obtenemos el valor de la altura dle rectangulo haciendo pop de la pila
157. alto = ((Double)pila.pop()).doubleValue();
158. **if**(g!=**null**){
159. (   **new** Rectangulo((**int**)X, (**int**)Y, (**int**)ancho, (**int**)alto )  ).dibuja(g);
160. }
161. }
163. **void** print(){
164. Double d;
165. d=(Double)pila.pop();
166. System.out.println(""+d.doubleValue());
167. }
169. **void** prexpr(){
170. Double d;
171. d=(Double)pila.pop();
172. System.out.print("["+d.doubleValue()+"]");
173. }
175. }

**Conclusiones.**

En esta práctica se pudo observar como YACC puede ser adaptado a distintos lenguajes como en este caso lo fue Java, esta práctica resulto ser más sencilla debido al hecho que al tener una mejor comprensión de YACC, solo fue de añadir más tokens e acciones gramaticales, además, de modificar un pequeño grupo de instrucciones en la máquina.