Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica número 2:

Uso de Java para modo gráfico

2 de enero de 2021

Grupo: 3CM7

Nombre del alumno:
Ramos Mesas Edgar Alain

Número de boleta:
2013090243

Materia: Compiladores

1. Introducción

Yacc y lex son herramientas de gran utilidad, compatibles y necesarias entre sí,para un diseñador de compiladores. Muchos compiladores se han construido utilizando estas herramientas (p.ej., el compilador de C de GNU (gcc)) o versiones más avanzadas. Los programas bison y flex son las versiones más modernas (no comerciales) de yacc y lex, y se distribuyen bajo licencia GPL con cualquier distribución de Linux (y también están disponibles para muchos otros UNIX). El programa lex genera analizadores léxicos a partir de una especificación de los componentes léxicos en términos de expresiones regulares (en el estilo de UNIX); lex toma como entrada un fichero (con la extensión.l) y produce un fichero en C (llamado "lex.yy.c") que contiene el analizador léxico. Yacc es un programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, . Otro generador de compiladores más". Genera un analizador sintáctico (la parte de un compilador que comprueba que la estructura del código fuente se ajusta a la especificación sintáctica del lenguaje) basado en una gramática analítica escrita en una notación similar a la BNF. Yacc fue desarrollado por Stephen C. Johnson en AT&T para el sistema operativo Unix. Después se escribieron programas compatibles, por ejemplo Berkeley Yacc, GNU bison, MKS yacc y Abraxas yacc. Cada una ofrece mejoras leves y características adicionales sobre el Yacc original, pero el concepto ha seguido siendo igual. Yacc también se ha reescrito para otros lenguajes, incluyendo Ratfor, EFL, ML, Ada y Java.
A pesar de que la mayoria de las aplicaciones de estas herramientas generalmente son poco asociadas con la interfa gráfica, si es posibleimplementar un modo gráfico en más de una aplicacion gracias a bibliotecas como las que proporciona Java o como las que pueden proporcionar cualquier otro lenguaje, que nos permitirá dibujar en pantalla diversas formas y colores.

2. Desarrollo

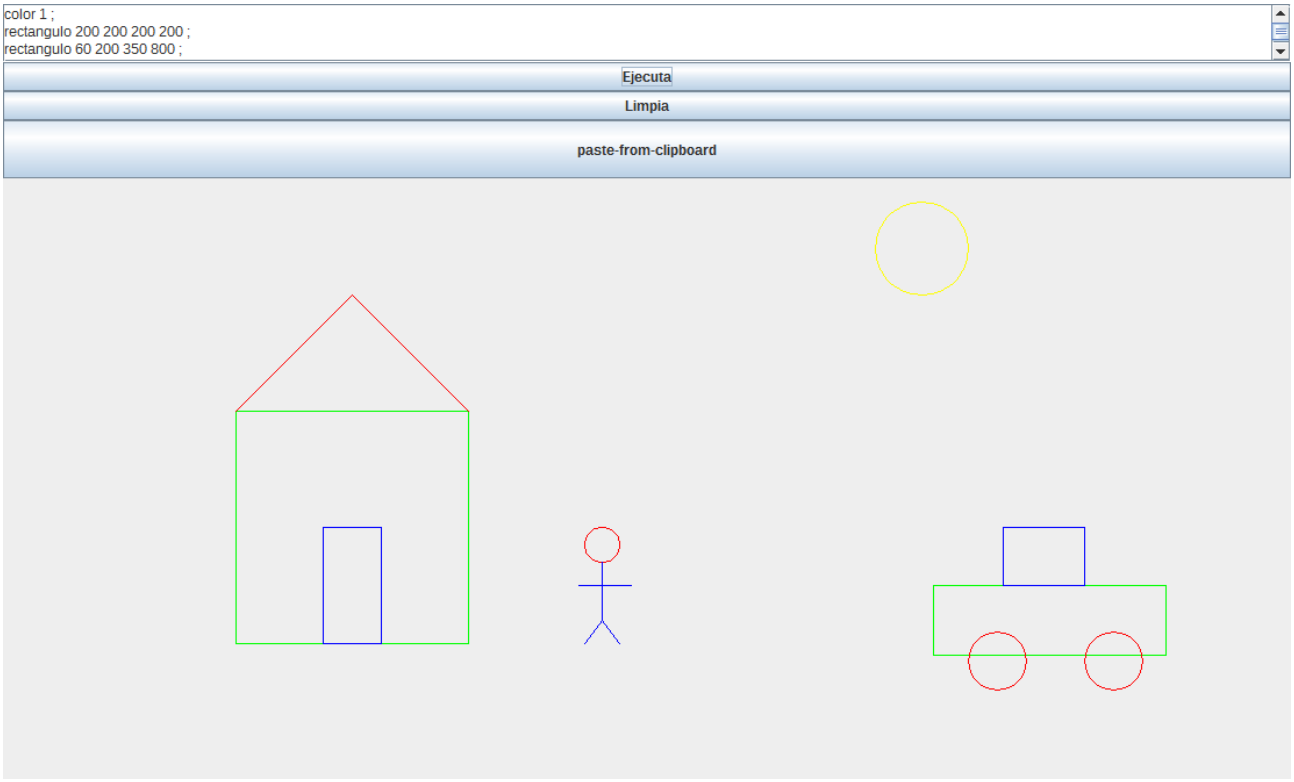
La práctica consistió básicamente en el uso de Yacc para desarrollar un programa que permita dibujar en un area especifica una casa, un sol y un coche, por lo cual fue necesario modificar diversos archivos pues los proporcionados por el profesor tenian aplicaciones distintas, y en principio solo podiamos dibujar lineas rectas, siendo esta figura solo una de las tres necesarias para poder realizar los dibujos.
Dado que tenemos las bases para dibujar lineas, resultó bastante intuitivo modificar los metodos para dibujar rectangulos y circulos, sin embargo, dentro de las modificaciones de los archivos la más destacable es la moudificación del archivo .y pues en el es en donde generamos las acciones gramaticales que serán ejecutadas. Como se puede observar en la siguiente imagen, se modifico la se sección de reglas grámaticales en el archivo para poder dibujar no solamente lineas sino que también las otras figuras haciendo uso de la maquina de pila. Ddo que en este caso estamos agregando las instrucciones a la pila se emplea la operación push con los tokens correspondientes.

```
| 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 {
   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("line");
   }

| CIRCULO NUMBER NUMBER NUMBER {
      maq.code("constpush");
      maq.code(((Algo)$2.obj).simb);
```

Además, se modificó el archivo "Maquina.java"para hacer la operación pop en la pila lo cual nos permitiío obtener los valores, mismos que se dibujaron después con su correspondiente método en Java. Se realizó lo mismo para las lineas y los circulos.

```
/**
    * Para nuestro caso un rectangulo está compuesto por:
    * X, Y, ancho, alto
 **/
void rectangulo(){
    double X, Y, ancho, alto;
    //Obtenemos el valor de la posición en X haciendo pop de la pila
    X = ((Double)pila.pop()).doubleValue();
    //Obtenemos el valor de la posición en Y haciendo pop de la pila
    Y = ((Double)pila.pop()).doubleValue();
    //Obtenemos el valor de la anchura del rectangulo haciendo pop de la pila
    ancho = ((Double)pila.pop()).doubleValue();
    //Obtenemos el valor de la altura dle rectangulo haciendo pop de la pila
    alto = ((Double)pila.pop()).doubleValue();
    if(g!=null){
        (   new Rectangulo((int)X, (int)Y, (int)ancho, (int)alto )  ).dibuja(g);
    }
}
```

Después de modificar el codigo y de escribir las instrucciones necesarias para realizar los dibujos solicitados se escribieron las instrucciones necesarias para dicha tarea, obteniendo finalmente al ejecutar el programa el siguiente resultado.



Además de lo ya mencionado y de las partes especificas que se han explicado del código, se agregan los códigos completos de los archivos modificados para el desarrollo de esta práctica siendo los más importantes los mostrados a continuación.

Archivo 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); }
22
23
24        | RECTANGULO NUMBER NUMBER NUMBER NUMBER  {
25          maq.code("constpush");
26          maq.code(((Algo)$2.obj).simb);
27
28          maq.code("constpush");
29          maq.code(((Algo)$3.obj).simb);
30
31          maq.code("constpush");
32          maq.code(((Algo)$4.obj).simb);
33
34          maq.code("constpush");
35          maq.code(((Algo)$5.obj).simb);
36
37          maq.code("rectangulo");
38          }
39
40        | LINE NUMBER NUMBER NUMBER NUMBER {
41          maq.code("constpush");
42          maq.code(((Algo)$2.obj).simb);
43
44          maq.code("constpush");
45          maq.code(((Algo)$3.obj).simb);
46
47          maq.code("constpush");
48          maq.code(((Algo)$4.obj).simb);
49
50          maq.code("constpush");
51          maq.code(((Algo)$5.obj).simb);
52
53          maq.code("line");
54          }
55
56        | CIRCULO NUMBER NUMBER NUMBER {
57              maq.code("constpush");
58              maq.code(((Algo)$2.obj).simb);
59
60              maq.code("constpush");
61              maq.code(((Algo)$3.obj).simb);
62
63              maq.code("constpush");
64              maq.code(((Algo)$4.obj).simb);
```

```
65
66              maq.code("circulo");}
67
68        | COLOR NUMBER { maq.code("constpush");
69                  maq.code(((Algo)$2.obj).simb); maq.code("color");}
70        ;
71  %%
72  class Algo {
73     Simbolo simb;
74     int inst;
75     public Algo(int i){ inst=i; }
76     public Algo(Simbolo s, int i){
77        simb=s; inst=i;
78     }
79  }
80  public void setTokenizer(StringTokenizer st){
81      this.st= st;
82  }
83  public void setNewline(boolean newline){
84      this.newline= newline;
85  }
86  Tabla tabla;
87  Maquina maq;
88
89  StringTokenizer st;
90  boolean newline;
91  int yylex(){
92  String s;
93  int tok;
94  Double d;
95  Simbolo simbo;
96     if (!st.hasMoreTokens())
97        if (!newline) {
98           newline=true;
99      return ';';
100       }
101    else
102       return 0;
103    s = st.nextToken();
104    try {
105       d = Double.valueOf(s);
106       yylval = new ParserVal(
107              new Algo(tabla.install("", NUMBER, d.doubleValue()),0) );
108       tok = NUMBER;
109    } catch (Exception e){
110    if(Character.isLetter(s.charAt(0))){
111       if((simbo=tabla.lookup(s))==null)
112          yylval = new ParserVal(new Algo(simbo, 0));
113      tok= simbo.tipo;
```

```
114       } else {
115        tok = s.charAt(0);
116       }
117    }
118    return tok;
119 }
120 void yyerror(String s){
121    System.out.println("parser error: "+s);
122 }
123 static Parser par = new Parser(0);
124 static JFrame jf;
125 static JLabel lmuestra=new JLabel("                                    ");
126 static Canvas canv;
127 static Graphics g;
128 Parser(int foo){
129    maq=new Maquina();
130    tabla=new Tabla();
131    tabla.install("line", LINE, 0.0);
132    tabla.install("circulo", CIRCULO, 0.0);
133    tabla.install("rectangulo", RECTANGULO, 0.0);
134    tabla.install("color", COLOR, 0.0);
135    tabla.install("print", PRINT, 0.0);
136    maq.setTabla(tabla);
137    jf=new JFrame("Calcula");
138    canv=new Canvas();
139    canv.setSize(600,600);
140    jf.add("North", new PanelEjecuta(maq, this));
141    jf.add("Center", canv);
142    jf.setSize( 600, 700);
143    jf.setVisible(true);
144    g=canv.getGraphics();
145    maq.setGraphics(g);
146    jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
147 }
148 public static void main(String args[]){ new Parser(); }
```

Archivo Maquina.java

```
1 import java.awt.*;
2 import java.util.*;
3 import java.lang.reflect.*;
4
5 class  Maquina {
6     Tabla tabla;
7     Stack pila;
8     Vector prog;
9
```

```
10      static int pc=0;
11      int progbase=0;
12      boolean returning=false;
13
14      Method metodo;
15      Method metodos[];
16      Class c;
17      Graphics g;
18      double angulo;
19      int x=0, y=0;
20      Class parames[];
21
22      Maquina(){
23      }
24
25      public void setTabla(Tabla t){
26          tabla = t;
27      }
28
29      public void setGraphics(Graphics g){
30          this.g=g;
31      }
32
33      Maquina(Graphics g){
34          this.g=g;
35      }
36
37      public Vector getProg(){
38          return prog;
39      }
40
41      void initcode(){
42          pila=new Stack();
43          prog=new Vector();
44      }
45
46      Object pop(){
47          return pila.pop();
48      }
49
50      int code(Object f){
51          System.out.println("Gen ("+f+") size="+prog.size());
52          prog.addElement(f);
53          return prog.size()-1;
54      }
55
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      }
85
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      }
93
94      void color(){
95          Color colors[]={Color.red,Color.green,Color.blue,Color.yellow};
96          double d1;
97          d1=((Double)pila.pop()).doubleValue();
98          if(g!=null){
99              g.setColor(colors[(int)d1]);
100         }
101     }
102
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();
117 
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     }
124 
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     }
143 
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     }
162 
163     void print(){
164         Double d;
165         d=(Double)pila.pop();
166         System.out.println(""+d.doubleValue());
167     }
168 
169     void prexpr(){
170         Double d;
171         d=(Double)pila.pop();
172         System.out.print("["+d.doubleValue()+"]");
173     }
174 
175 }
```

3. Conclusiones

En esta práctica se pudo observar como YACC puede ser adaptado en distintos lenguajes como en este caso lo fue Java, además de permitirnos no solo diseñar aplicaciones en modo consola sino que también nos permite generar programas que hagan uso de herrameintas graficas.
Esta práctica resulto ser más sencilla que la primera debido al hecho que al tener una mejor comprensión de YACC, solo fue necesario añadir más tokens y acciones gramaticales, además, de modificar un pequeño grupo de instrucciones en la máquina con el fin de poder dibujar las figuras solicitadas en la práctica, probando así que Yacc no se limita solo aplicaciones en modo texto, sino que también permite aplicaciones gráficas.