INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Nombre: Cruz López Adrián

Grupo: 3CM15

Asignatura: Compiladores

Profesor: Roberto Tecla Parra

Actividad: **Práctica 2 Uso de Java para modo gráfico** “Para dibujar círculos, líneas y rectángulos”

Fecha: 27/10/2021

INTRODUCCIÓN

YACC

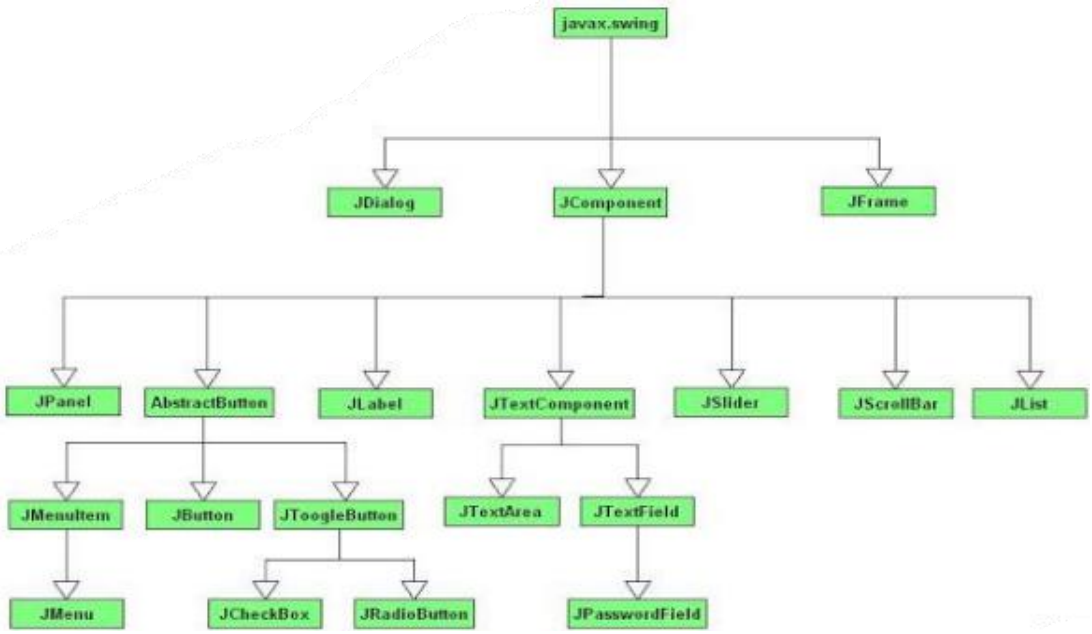
Es un programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, "Otro generador de compiladores más". Genera un analizador sintáctico (la parte de un compilador que comprueba que la estructura del código fuente se ajusta a la especificación sintáctica del lenguaje) basado en una gramática analítica escrita en una notación similar a la BNF. YACC genera el código para el analizador sintáctico en el Lenguaje de programación C.

Puesto que el analizador sintáctico generado por YACC requiere un analizador léxico, se utiliza a menudo juntamente con un generador de analizador léxico, en la mayoría de los casos lex o Flex, alternativa del software libre. El estándar de IEEE POSIX P1003.2 define la funcionalidad y los requisitos a Lex y YACC.

Java en modo gráfico

Swing

Es un conjunto de clases y otros recursos para construir GUI's (Interfaces de Usuario Gráficas). Es independiente de la plataforma (sistema operativo) sobre la que se ejecute. Con Swing podemos crear ventajas, contenedores, botones, etiquetas, campos de texto, listas desplegables, etc. La mayoría de sus clases comienzan con una J (por ejemplo, JFrame). La librería java.swing es una librería más estándar ya que esta no depende del SO, es decir que permite una interfaz a cada SO sin cambio de código. Los componentes de la librería son:

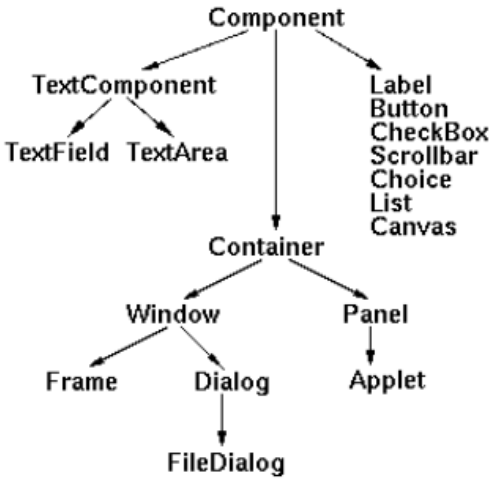


AWT

Significa Abstract Window Toolkit. Es dependiente de la plataforma, lo que quiere decir que la imagen que muestre un componente gráfico dependerá del sistema operativo en el que se ejecute.

Jerarquía de clases de AWT

Cada una de las componentes de una ventana en AWT se representa mediante uno o más objetos de la aplicación. Estos objetos pertenecen a las clases que se observan en la siguiente jerarquía de clases para AWT:



DESARROLLO

Utilizando los archivos de la carpeta *grafibasi1,* y ya que se elegio la opción de dibujar círculos, líneas y rectángulos, la gramática es la siguiente:

```
%%
        list : /*Nada (epsilon)*/
        | list ';'
        | list inst ';'
        ;
        inst: NUMBER
            | RECTANGULO NUMBER NUMBER NUMBER NUMBER
            | LINE NUMBER NUMBER NUMBER NUMBER
            | CIRCULO NUMBER NUMBER NUMBER
            | COLOR NUMBER
            ;
%%
```

Es equivalente a:

list → ε
list → list ;
list → list inst ;
inst → NUMBER
inst → RECTANGULO NUMBER NUMBER NUMBER NUMBER
inst → LINE NUMBER NUMBER NUMBER NUMBER
inst → CIRCULO NUMBER NUMBER NUMBER
inst → COLOR NUMBER

Una sintaxis más familiar que cubre los casos del código que acepta el programa es:

rectangulo 20 20 20 20;
circulo 30 30 50;
line 30 30 30 30;
color 2;

Cada uno recibe una cantidad determinada de valores, los cuales sirven para determinar las medidas de las figuras.

El archivo *rectangulo.java*, como su nombre lo indica, sirve para el caso de que se tenga que dibujar un rectángulo, el código de dicho archivo es el siguiente:

```
19. public Rectangulo(int x1, int y1, int x2, int y2){
20.       this.x1=x1;
21.       this.y1=y1;
22.       this.x2=x2;
23.       this.y2=y2;
24. }
25.
26. public void dibuja(Graphics g){
27.       g.drawRect(x1,y1,x2,y2);
28. }
```

Se puede observar que se reciben 4 valores, los cuales son:

- X1 y Y1; el punto de origen
- X2; el ancho
- Y2; su altura

Posteriormente se mandan los parámetros a la función *drawRect(),* la cual únicamente dibuja el contorno de dicho rectángulo, en caso de quererlo pintado, se llama a la función *fillRect().*

Para el caso de dibujar la línea es algo muy similar, este código se encuentra dentro del archivo *Linea.java:*

```
19. public Linea(int x1, int y1, int x2, int y2){
20.     this.x1=x1;
21.     this.y1=y1;
22.     this.x2=x2;
23.     this.y2=y2;
24. }
25. public void dibuja(Graphics g){
26.     System.out.println("en dib linea");
27.     g.drawLine(x1,y1,x2,y2);
28. }
```

En este caso de igual manera, se pasan 4 valores, los cuales corresponden a:

- X1 y Y1; la posición inicial de la línea
- X2 y Y2; la posición final de la línea

Posteriormente se pasan a la función *drawLine(),* la cual dibujará la línea con del tamaño que se le indique.

Finalmente, el caso del círculo, cuyo código está en el archivo *circulo.java*, una parte del contenido de dicho archivo es el siguiente:

```
17. public Circulo(int x, int y, int r){
18.     this.x=x;
19.     this.y=y;
20.     this.r=r;
21. }
22.
23. public void dibuja(Graphics g){
24.     g.drawOval(x,y,r,r);
25. }
```

En este caso mandan 4 valores a la función *drawOval(),* la cual se encargará de dibujar el círculo:

- X y Y; las coordenadas iniciales del circulo
- Los últimos 2 valores son el radio, esto para que el ovalo quede en forma de círculo. Debido a que, en esta función, estos valores representan la altura y el ancho del ovalo.

Para el caso del archivo de *Maquina.java*, se encuentran las funciones que sirven para sacar los valores de la pila, valores con los que se van a dibujar las distintas figuras aceptadas. La siguiente función que se muestra es la del rectángulo:

```
157. void rectangulo(){
158.     double X, Y, ancho, alto;
159.     //Se obtine el valor de la posicion en X haciendo pop de la pila
160.     X = ((Double)pila.pop()).doubleValue();
161.     // Se obtine  el valor de la posicion en Y haciendo pop de la pila
162.     Y = ((Double)pila.pop()).doubleValue();
163.   // Se obtine el valor de la anchura del rectangulo haciendo pop de la pila
164.     ancho = ((Double)pila.pop()).doubleValue();
165.   // Se obtine el valor de la altura del rectangulo haciendo pop de la pila
166.     alto = ((Double)pila.pop()).doubleValue();
167.     if(g!=null){
168.       (new Rectangulo((int)X, (int)Y, (int)ancho, (int)alto )  ).dibuja(g);
169.     }
170. }
```

Se realiza un pop a los valores contenidos en la pila, para poder obtenerlos y realizar los procedimientos correspondientes.

Finalmente, dentro del archivo *Tabla.java*, se tienen las funciones de *install* y *lookup,* las cuales sirven al momento de leer el código escrito en el programa, de manera que las palabras clave se guarden ahí.

```
16. Simbolo install(String s, short t, double d){
17.         Simbolo simb=new Simbolo(s,t,d);
18.         simb.ponSig(listaSimbolo);
19.         listaSimbolo=simb;
20.         return simb;
21.     }
22.
23. Simbolo lookup(String s){
24.     for(Simbolo sp=listaSimbolo; sp!=null; sp=sp.obtenSig())
25.         if((sp.obtenNombre()).equals(s))
26.             return sp;
27.     return null;
28.     }
29. }
```

PRUEBAS DE FUNCIONAMIENTO

Circulo

El primer y segundo valor representa la posición inicial, mientras que el tercer valor corresponde al radio, y el que hace que el ovalo tome forma de circulo.

Rectángulo

Los primeros 2 valores representan el punto de origen, mientras que el tercer valor corresponde al ancho y el cuarto es el valor de la altura del rectángulo.

Línea

En el último caso de la línea, los dos primeros valores corresponden al punto de origen, mientras que los últimos 2 corresponden al punto final, de esta manera se traza la línea.

En los 3 casos mostrados se utilizó la sintaxis mostrada en un principio, y como se puede observar, al ingresarle los valores que estén dentro del rango de dicha ventana, mostrará la figura correspondiente

CONCLUSIÓN

Al realizar esta práctica, pude utilizar los conocimientos visto hasta ahora durante las clases de este curso sobre YACC básico y del HOC1. Pude poner en práctica lo visto acerca de la sintaxis que se utiliza en YACC para definir la gramática, las acciones gramaticales, etc.

De igual manera, pude aplicar estos conocimientos, además de algunos de los vistos en materias anteriores como es el caso de Programación Orientada a Objetos, en la que se hizo uso de herramientas gráficas que nos proporciona Java, a diferencia de la práctica 1, en esta práctica se trabajó con el lenguaje Java, comprobando que se pueden utilizar cualquiera de estos 2 lenguajes de programación (C y Java).

REFERENCIAS

S. Federico. (s.f.). El generador de analizadores sintácticos yacc. Recuperado 22 de octubre de 2021, de https://www.infor.uva.es/~mluisa/talf/docs/labo/L8.pdf

Naps. (2017, 16 noviembre). Modo gráfico en Java: Creando un lienzo propio -. Recuperado 22 de octubre de 2021, de https://naps.com.mx/blog/modo-grafico-java-creando-lienzo-propio/-p

El Lenguaje Java. (s. f.). Recuperado 22 de octubre de 2021, de https://users.dcc.uchile.cl/%7Elmateu/Java/Apuntes/awt.htm