Instituto Politécnico Nacional

Escuela Superior de Cómputo

Compiladores

“Proyecto de Compiladores: Logo”

Manual Técnico

Grupo: 3CM7

Integrantes:

- Martínez Coronel Brayan Yosafat.
- Ramírez Olvera Guillermo.
- Sánchez Méndez Edmundo Josue.

Fecha de entrega: 18 de enero de 2021

Profesor: Tecla Parra Roberto

ESCOM

INTRODUCCIÓN

LOGO es un lenguaje de programación de alto nivel es, en parte funcional, en parte estructurado; de muy fácil aprendizaje, razón por la cual suele ser el lenguaje de programación preferido para trabajar con niños y jóvenes. Fue diseñado con nes didácticos por Danny Bobrow, Wally Feurzeig, Seymour Papert y Cynthia Solomon, los cuales se basaron en las características del lenguaje Lisp. Logo fue creado con la finalidad de usarlo para enseñar programación y puede usarse para enseñar la mayoría de los principales conceptos de la programación, ya que proporciona soporte para manejo de listas, archivos y entrada/salida. Logo cuenta con varias versiones. Una característica más explotada de Logo es poder producir <<gráficos tortuga>>, es decir, poder dar instrucciones a una tortuga virtual, mediante un cursor grafico usado para crear dibujos, que en algunas versiones es un triángulo, en otras tiene la figura de una tortuga vista desde arriba. Esta tortuga o cursor se maneja mediante palabras que representan instrucciones. El desarrollo de este proyecto consistirá en una interfaz gráfica para el usuario con un apartado en donde él pueda agregar código con la finalidad de dibujar desde las figuras más sencillas como cuadrados y círculos, hasta espirales, arboles, estrellas, etc. Con la posibilidad de agregar colores a las figuras dibujadas y levantar el pincel para poder moverlo a voluntad sin dibujar nada y también de bajar el pincel, para empezar a dibujar nuevamente, esto se mostrará en una vista en el mismo programa. Para el desarrollo de la interfaz gráfica se cuenta con la API de JAVA:AWT, para aplicaciones de escritorio en Windows/Linux.

OBJETIVO

El objetivo de este trabajo es dar una referencia técnica acerca del proyecto para que pueda ser entendido por alguna persona con los conocimientos técnicos aptos y así poder ser modificado, mejorado o alguna otra acción en el software.

REQUERIMIENTOS TÉCNICOS

Java

Plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarían a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y viable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. Java es la base para prácticamente todos los tipos de aplicaciones de redes, ademas del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en web y software de empresa. Con más de 9 millones de desarrolladores en todo el mundo, Java le permite desarrollar, implementar y utilizar de forma eficaz interesantes aplicaciones y servicios. Java está en todas partes.

Java Development Kit (JDK)

El JDK es un entorno de desarrollo para crear aplicaciones, applets y componentes utilizando el lenguaje de programación Java. El JDK incluye

herramientas útiles para desarrollar y probar programas escritos en el lenguaje de programación Java y que se ejecutan en la plataforma Java.

Java Runtime Edition (JRE)

Java Runtime Environment (JRE) es lo que se obtiene al descargar el software de Java. JRE está formado por Java Virtual Machine(JVM), clases del núcleo de la plataforma Java y bibliotecas de la plataforma Java de soporte. JRE es la parte de tiempo de ejecución del software de Java, que es todo lo que necesita para ejecutarlo en el explorador web.

BYACC

El cual es un compilador para el lenguaje YACC, pero enfocado al desarrollo en Java.

EJECUCION

Lo primero que vamos a hacer será buscar el IDE de NetBeans y abrirlo en nuestro escritorio. Una vez que abrimos el IDE nos aparecerá la pantalla de inicio en la cual tendremos varias opciones, pero la que es de nuestro interés es la opción que dice Abrir Proyecto o “Open Proyect”, esto para poder abrir nuestro proyecto.

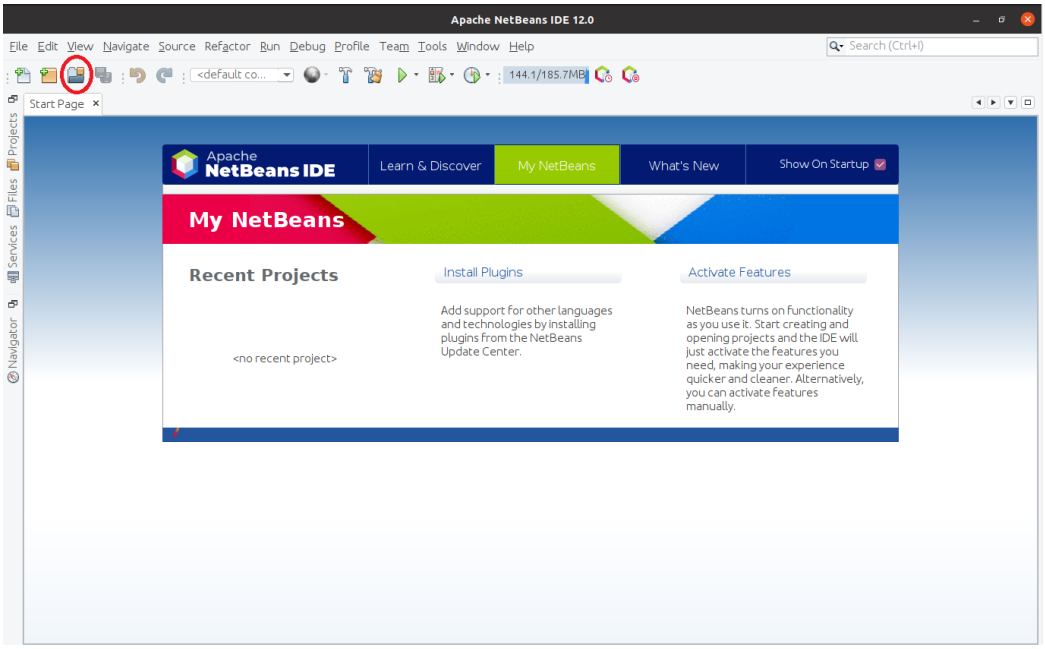


Imagen 1. NetBeans abierto por primera vez.

Ahora se nos abrirá el buscador de archivos, esto para poder buscar la carpeta del proyecto que lleva por nombre “ProyectoLogos”, una vez encontrada procederemos a abrir el proyecto y esto abrir el proyecto en NetBeans.

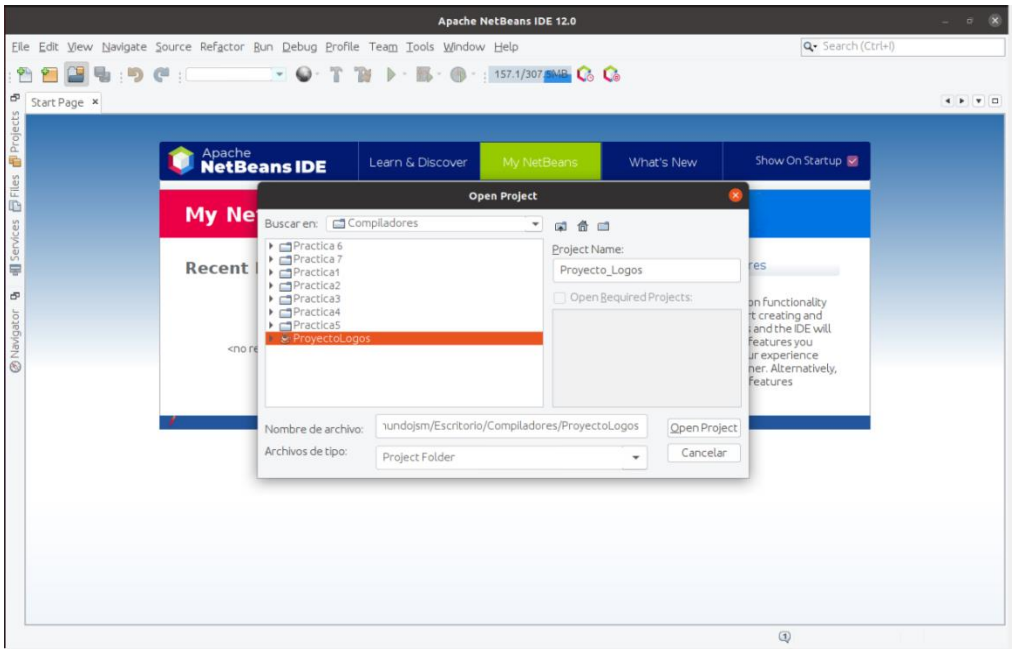


Imagen 2. Selección del proyecto para poder abrirlo en NetBeans.

Una vez abierto el proyecto por defecto se nos abrirá en la sección “Projects” de ahí abriremos la carpeta llamada “Source Packages”, de ahí nos pasaremos a la subcarpeta “logos” y encontraremos un archivo llamado Logos.java, como guía ver la imagen y en especial el cuadrado rojo.

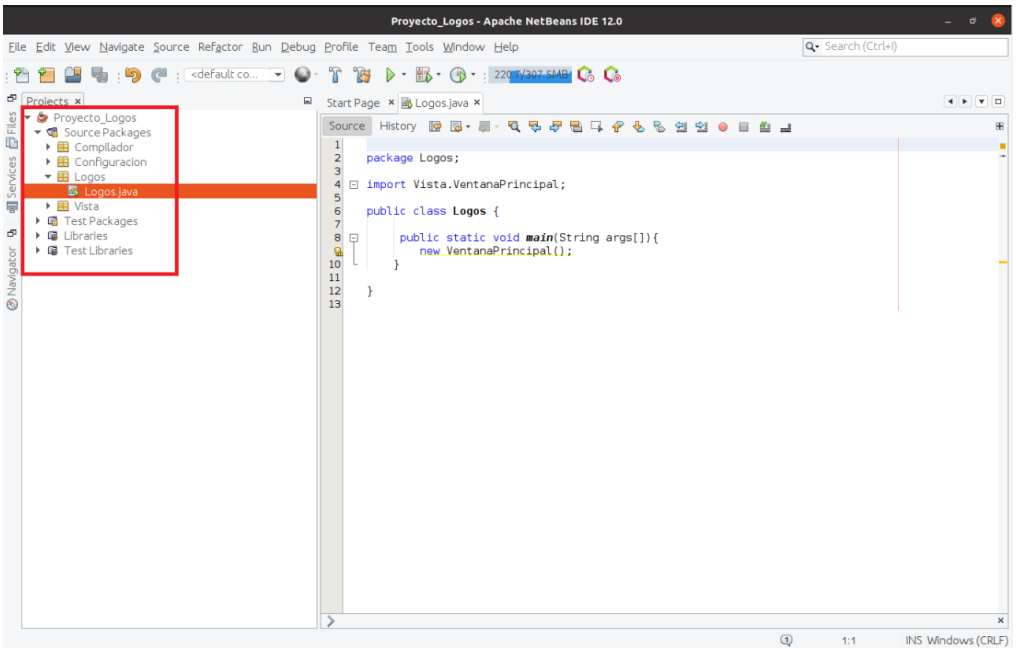


Imagen 3. Selección del archivo Logos.java.

Ahora procederemos a ejecutar el programa, para hacer esto tenemos dos formas la primera es dar clic derecho y se nos desplegara un menú en donde seleccionaremos “Run File” y así ejecutaremos el proyecto (para una guía más

visual dirigirse la Imagen 4), la otra forma es dar clic en la flecha de color verde que tenemos en el menú de herramientas, el cual también nos ejecutara , el archivo nos abrirá una interfaz gráfica en la cual podemos interactuar con el programa.

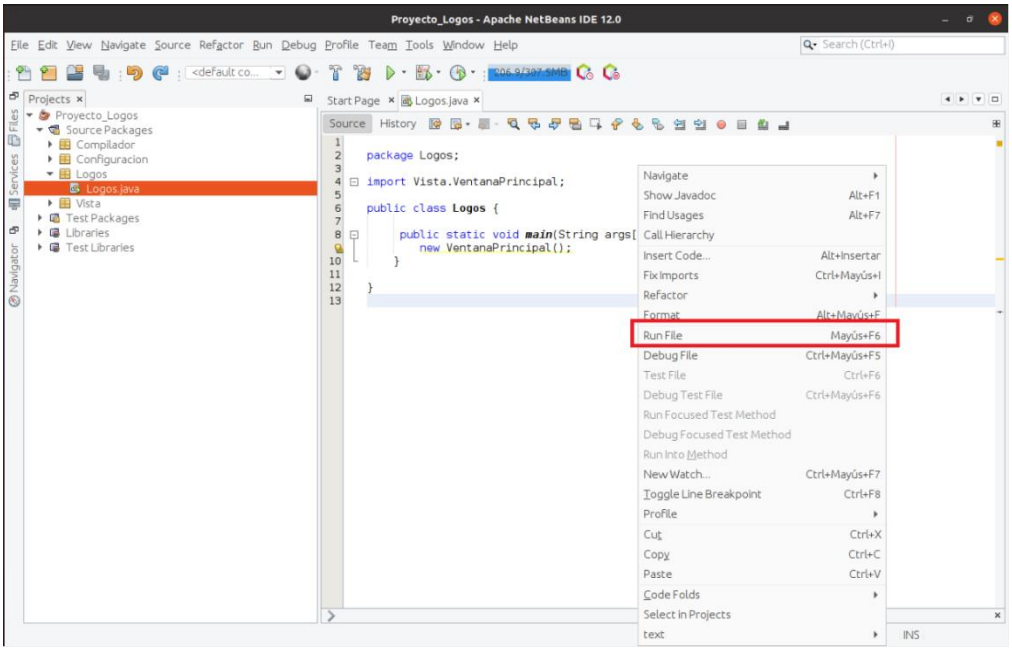


Imagen 4. Ejecución del proyecto ejecutando el archivo Logos.java.

Una vez que se ejecuta el programa tendremos la interfaz gráfica mostrada en la Imagen 5, en donde tenemos los siguientes componentes:

- Un área de texto, donde se ingresan los comandos del lenguaje.
- Un botón que permite dibujar las acciones descritas por los comandos.
- Un botón que permite borrar lo que esta dibujado en el panel y nos permite realizar el botón del siguiente punto, si fue seleccionado anteriormente y se le vuelve a seleccionar regresamos se nos deshabilita la opción del botón siguiente y guarda el estado en el panel lo que hicimos mientras lo seleccionamos por primera vez.
- Un botón que dibuja paso a paso la figura.
- Un panel, donde se visualiza el dibujo.

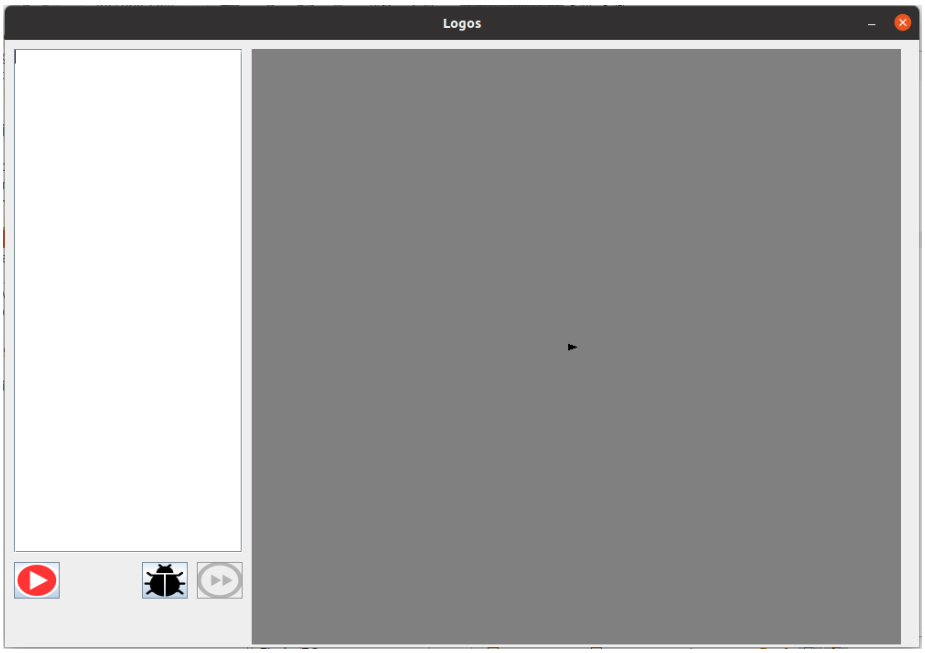


Imagen 5. Interfaz gráfica de usuario.

EXPRESIONES REGULARES

Las siguientes son expresiones regulares:

- FORWARD[double];
- TURN[double];
- COLOR[double, double, double];
- PenUP[];
- PenDOWN[];
- for(exploración inicial; condición; expresión final){<bloque>}
- while(condición){ declaracion(es)}
- if (condición){instrucciones}else{instrucciones}
  proc [nombre](){Bloque de instrucciones} **Para usar parámetros usar $n, donde n es el parámetro para usar**, en las llamadas los parámetros se separan con una ‘,’.
- func [nombre](){
  Bloque de instrucciones
  return valor;
  } **Para usar parámetros usar $n, donde n es el parámetro para usar**, en las llamadas los parámetros se separan con una ‘,’.
  Condiciones disponibles:
- ==, comparación, entre enteros
- !=, diferente, entre enteros
- <, operación menor o igual, entre enteros.
- <=, operación menor o igual, entre enteros.
- >, operación mayor, entre enteros.
- >=, operación mayor o igual, entre enteros.

- &&, operación “and”, entre enteros.
  ||, operación “or”, entre enteros.

GRAMATICA

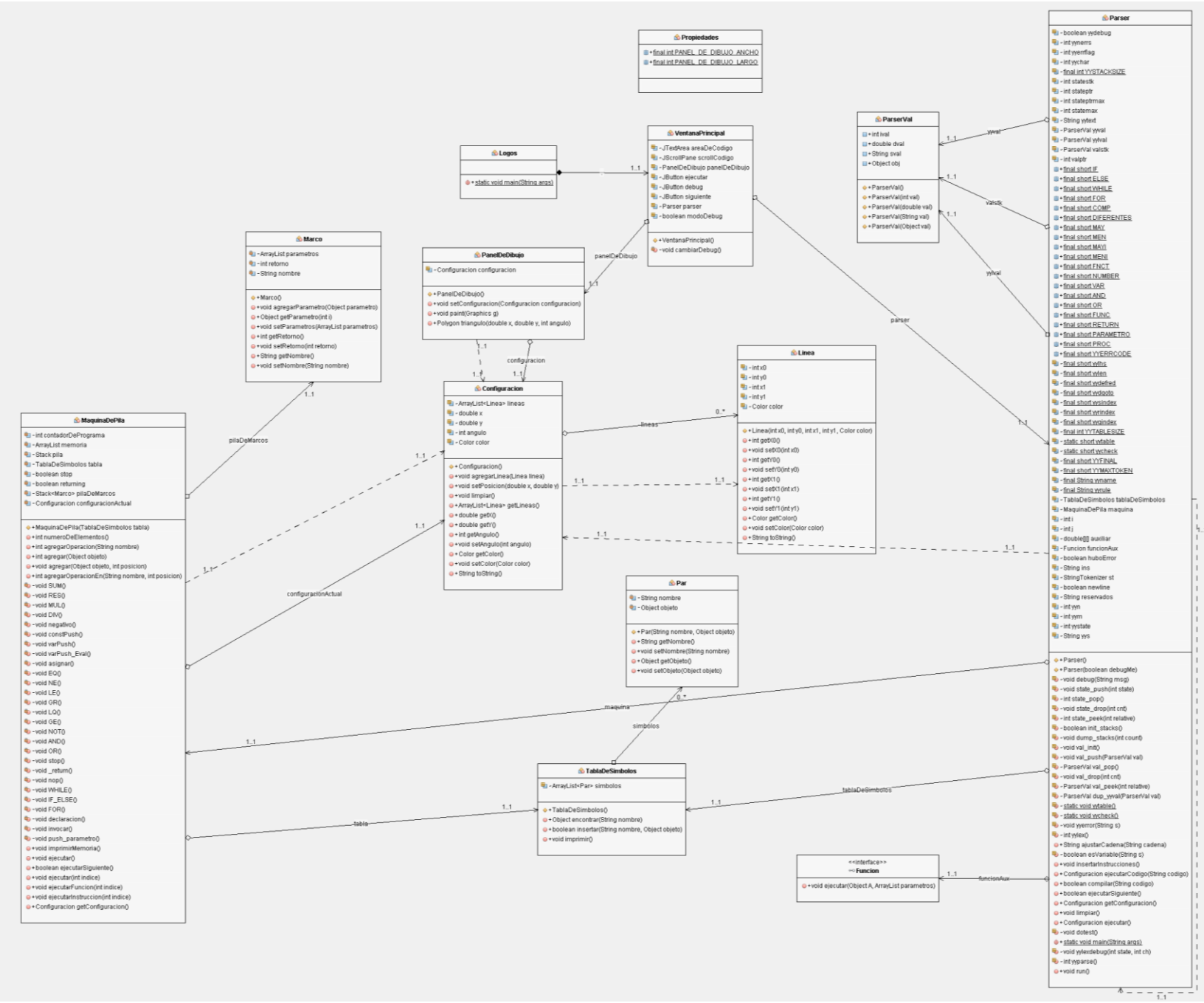
```
%token IF
%token ELSE
%token WHILE
%token FOR
%token COMP
%token DIFERENTES
%token MAY
%token MEN
%token MAYI
%token MENI
%token FNCT
%token NUMBER
%token VAR
%token AND
%token OR
%token FUNC
%token RETURN
%token PARAMETRO
%token PROC
%right '='
%left '+' '-'
%left '*' '/'
%left ';'
%left COMP
%left DIFERENTES
%left MAY
%left MAYI
%left MEN
%left MENI
%left '!'
%left AND
%left OR
%right RETURN
%%

    list:
        | list'\n'
        | list linea '\n'
        ;

    linea: exp ';' {$$ = $1;}
        |stmt {$$ = $1;}
        |linea exp ';' {$$ = $1;}
        |linea stmt {$$ = $1;}
        ;

    exp:  VAR {
                $$ = new ParserVal(maquina.agregarOperacion("varPush_Eval
"));
                maquina.agregar($1.sval);
            }
        |'-' exp {
```

```
                $$ = new ParserVal(maquina.agregarOperacion("negativo"));
            }
        |NUMBER {
                $$ = new ParserVal(maquina.agregarOperacion("constPush"))
;
                maquina.agregar($1.dval);
            }
        | VAR '=' exp {
                $$ = new ParserVal($3.ival);
                maquina.agregarOperacion("varPush");
                maquina.agregar($1.sval);
                maquina.agregarOperacion("asignar");
                maquina.agregarOperacion("varPush_Eval");
                maquina.agregar($1.sval);
            }
        | exp '*' exp {
                $$ = new ParserVal($1.ival);
                maquina.agregarOperacion("MUL");
            }
                | exp '/' exp {
                $$ = new ParserVal($1.ival);
                maquina.agregarOperacion("DIV");
            }
        | exp '+' exp {
                $$ = new ParserVal($1.ival);
                maquina.agregarOperacion("SUM");
            }
        | exp '-' exp {
                $$ = new ParserVal($1.ival);
                maquina.agregarOperacion("RES");
            }
        |'(' exp ')' {
                $$ = new ParserVal($2.ival);
            }
        | exp COMP exp {
                 maquina.agregarOperacion("EQ");
                 $$ = $1;
            }
        | exp DIFERENTES exp {
                 maquina.agregarOperacion("NE");
                 $$ = $1;
            }
        | exp MEN exp {
                 maquina.agregarOperacion("LE");
                 $$ = $1;
            }
        | exp MENI exp {
                 maquina.agregarOperacion("LQ");
                 $$ = $1;
            }
        | exp MAY exp {
                 maquina.agregarOperacion("GR");
                 $$ = $1;
            }
        | exp MAYI exp {
                 maquina.agregarOperacion("GE");
                 $$ = $1;
```

```
            }
        | exp AND exp {
                maquina.agregarOperacion("AND");
                 $$ = $1;
            }
        | exp OR exp {
                maquina.agregarOperacion("OR");
                 $$ = $1;
            }
        | '!' exp {
                maquina.agregarOperacion("NOT");
                $$ = $2;
            }
        | RETURN exp { $$ = $2; maquina.agregarOperacion("_return"); }

        | PARAMETRO { $$ = new ParserVal(maquina.agregarOperacion("push_p
arametro")); maquina.agregar((int)$1.ival); }

        |nombreProc '(' arglist ')' { $$ = new ParserVal(maquina.agregarO
peracionEn("invocar",($1.ival))); maquina.agregar(null); } //instruccione
s tiene la estructura necesaria para la lista de argumentos
        ;

    arglist:
        |exp {$$ = $1; maquina.agregar("Limite");}
        |arglist ',' exp {$$ = $1; maquina.agregar("Limite");}
        ;

    nop: {$$ = new ParserVal(maquina.agregarOperacion("nop"));}
        ;

    stmt:if '(' exp stop ')' '{' linea stop '}' ELSE '{' linea stop'}' {
                $$ = $1;
                maquina.agregar($7.ival, $1.ival + 1);
                maquina.agregar($12.ival, $1.ival + 2);
                maquina.agregar(maquina.numeroDeElementos() - 1, $1.ival
+ 3);
            }
        | if '(' exp stop ')' '{' linea stop '}' nop stop{
                $$ = $1;
                maquina.agregar($7.ival, $1.ival + 1);
                maquina.agregar($10.ival, $1.ival + 2);
                maquina.agregar(maquina.numeroDeElementos() - 1, $1.ival
+ 3);
            }
        | while '(' exp stop ')' '{' linea stop '}' stop{
                $$ = $1;
                maquina.agregar($7.ival, $1.ival + 1);
                maquina.agregar($10.ival, $1.ival + 2);
            }
        | for '(' instrucciones stop ';' exp stop ';' instrucciones stop
')' '{' linea stop '}' stop{
                $$ = $1;
                maquina.agregar($6.ival, $1.ival + 1);
                maquina.agregar($9.ival, $1.ival + 2);
                maquina.agregar($13.ival, $1.ival + 3);
                maquina.agregar($16.ival, $1.ival + 4);
```

```
            }
        | funcion nombreProc '(' ')' '{' linea null '}'
        | procedimiento nombreProc '(' ')' '{' linea null '}'
        | instruccion '[' arglist ']' ';' {
                $$ = new ParserVal($1.ival);
                maquina.agregar(null);
            }
        ;
    instruccion: FNCT {
            $$ = new ParserVal(maquina.agregar((Funcion)($1.obj)));
            }
        ;

    procedimiento: PROC { maquina.agregarOperacion("declaracion"); }
        ;
    funcion: FUNC { maquina.agregarOperacion("declaracion"); }
        ;

    nombreProc: VAR {$$ = new ParserVal(maquina.agregar($1.sval));}
        ;

    null: {maquina.agregar(null);}
        ;

    stop: {$$ = new ParserVal(maquina.agregarOperacion("stop"));}
        ;

    if: IF {
            $$ = new ParserVal(maquina.agregarOperacion("IF_ELSE"));
            maquina.agregarOperacion("stop");//then
            maquina.agregarOperacion("stop");//else
            maquina.agregarOperacion("stop");//siguiente comando
        }
        ;

    while: WHILE {
            $$ = new ParserVal(maquina.agregarOperacion("WHILE"));
            maquina.agregarOperacion("stop");//cuerpo
            maquina.agregarOperacion("stop");//final
        }
        ;

    for : FOR {
            $$ = new ParserVal(maquina.agregarOperacion("FOR"));
            maquina.agregarOperacion("stop");//condicion
            maquina.agregarOperacion("stop");//instrucción final
            maquina.agregarOperacion("stop");//cuerpo
            maquina.agregarOperacion("stop");//final
        }

    instrucciones: { $$ = new ParserVal(maquina.agregarOperacion("nop"));
}
        |exp {$$ = $1;}
        |instrucciones ',' exp {$$ = $1;}
        ;
%%
```

DIAGRAMA DE CLASES



EXPLICACION DE ALTO NIVEL

Para la realización de este proyecto fue necesario hacer uso de todos los conocimientos aprendidos en esta materia, para ello lo primero es hacer nuestra clase Par la cual es como las que hemos usado en anteriores ocasiones para las prácticas en este caso tendremos un nombre y un objeto, en segunda necesitamos nuestra tabla de símbolos la cual contendrá todos nuestras instrucciones las cuales son: TURN, FORWARD, COLOR, PenUP y PenDOWN, después necesitamos nuestra máquina de pila, la cual se encuentra en el archivo Java que posee el mismo nombre que es sin duda alguna una de las partes más esenciales de nuestro proyecto ya que es ahí donde sucede todo ya que tiene el funcionamiento de las condiciones, de las operaciones que podemos hacer y por supuesto de nuestras instrucciones el código es el siguiente:

```
    private void declaracion(){
        tabla.insertar((String)memoria.get(++contadorDePrograma), ++conta
dorDePrograma); //Apuntamos a la primera instrucción de la función
        int invocados = 0;
        while(memoria.get(contadorDePrograma) != null || invocados != 0){
 //Llevamos cp hasta la siguiente instrucción después de la declaración
            if( memoria.get(contadorDePrograma) instanceof Method)
                if(((Method)memoria.get(contadorDePrograma)).getName().eq
uals("invocar"))
                    invocados++;
            if( memoria.get(contadorDePrograma) instanceof Funcion)
                invocados++;
            if(memoria.get(contadorDePrograma) == null)
                invocados--;
            contadorDePrograma++;
        }
    }

    private void invocar(){
        Marco marco = new Marco();
        String nombre = (String)memoria.get(++contadorDePrograma);
        marco.setNombre(nombre);
        contadorDePrograma++;
        while(memoria.get(contadorDePrograma) != null){ //Aquí también us
amos null como delimitador. Aquí se agregan los parámetros al marco
            if(memoria.get(contadorDePrograma) instanceof String){
                if(((String)(memoria.get(contadorDePrograma))).equals("Li
mite")){
                    Object parametro = pila.pop();
                    marco.agregarParametro(parametro);
                    contadorDePrograma++;
                }
            }
            else{
                ejecutarInstruccion(contadorDePrograma);
            }

        }
        marco.setRetorno(contadorDePrograma);
        pilaDeMarcos.add(marco);
```

```
        ejecutarFuncion((int)tabla.encontrar(nombre)); //VAMOS AQUI******
**********
    }

    private void push_parametro(){
        pila.push(pilaDeMarcos.lastElement().getParametro((int)memoria.ge
t(++contadorDePrograma)-1));
    }

    public void ejecutar(){
        //imprimirMemoria();
        stop = false;
        while(contadorDePrograma < memoria.size())
            ejecutarInstruccion(contadorDePrograma);
    }

    public boolean ejecutarSiguiente(){
        //imprimirMemoria();
        if(contadorDePrograma < memoria.size()){
            ejecutarInstruccion(contadorDePrograma);
            return true;
        }
        return false;
    }

    public void ejecutar(int indice){//ejecuta hasta que se encuentra Sto
p
        contadorDePrograma = indice;
        while(!stop && !returning){
            ejecutarInstruccion(contadorDePrograma);
        }
        stop = false;
    }

    public void ejecutarFuncion(int indice){
        contadorDePrograma = indice;
        while(!returning && memoria.get(contadorDePrograma) != null){
            ejecutarInstruccion(contadorDePrograma);
        }
        returning = false;
        contadorDePrograma = pilaDeMarcos.lastElement().getRetorno();
        pilaDeMarcos.removeElement(pilaDeMarcos.lastElement());
    }

    public void ejecutarInstruccion(int indice){
        //System.out.println("Ejecutando: " + indice);
        try{
            Object objetoLeido = memoria.get(indice);
            if(objetoLeido instanceof Method){
                Method metodo = (Method)objetoLeido;
                metodo.invoke(this, null);
            }
            if(objetoLeido instanceof Funcion){
                ArrayList parametros = new ArrayList();
                Funcion funcion = (Funcion)objetoLeido;
                contadorDePrograma++;
```

```
                while(memoria.get(contadorDePrograma) != null){ //Aquí ta
mbién usamos null como delimitador. Aquí se agregan los parámetros al mar
co
                    if(memoria.get(contadorDePrograma) instanceof String)
{
                        if(((String)(memoria.get(contadorDePrograma))).eq
uals("Limite")){
                            Object parametro = pila.pop();
                            parametros.add(parametro);
                            contadorDePrograma++;
                        }
                    }
                    else{
                        ejecutarInstruccion(contadorDePrograma);
                    }

                }
                funcion.ejecutar(configuracionActual, parametros);
            }
            contadorDePrograma++;
        }
        catch(Exception e){}
    }

    public Configuracion getConfiguracion(){
        return configuracionActual;
    }

    public static class Girar implements Funcion{
        @Override
        public void ejecutar(Object A, ArrayList parametros) {
            Configuracion configuracion = (Configuracion)A;
            int angulo = (configuracion.getAngulo() + (int)(double)parame
tros.get(0))%360;
            configuracion.setAngulo(angulo);
        }
    }

    public static class Avanzar implements Funcion{
        @Override
        public void ejecutar(Object A, ArrayList parametros) {
            Configuracion configuracion = (Configuracion)A;
            int angulo = configuracion.getAngulo();
            double x0 = configuracion.getX();
            double y0 = configuracion.getY();
            double x1 = x0 + Math.cos(Math.toRadians(angulo))*(double)par
ametros.get(0);
            double y1 = y0 + Math.sin(Math.toRadians(angulo))*(double)par
ametros.get(0);
            configuracion.setPosicion(x1, y1);
            configuracion.agregarLinea(new Linea((int)x0,(int)y0,(int)x1,
(int)y1, configuracion.getColor()));
        }
    }

    public static class CambiarColor implements Funcion{
        @Override
```

```
        public void ejecutar(Object A, ArrayList parametros) {
            Configuracion configuracion = (Configuracion)A;
            configuracion.setColor(new Color((int)(double)parametros.get(
0)%256, (int)(double)parametros.get(1)%256, (int)(double)parametros.get(2
)%256));
        }
    }

    public static class SubirPincel implements Funcion{
        @Override
        public void ejecutar(Object A, ArrayList parametros) {
            Configuracion configuracion = (Configuracion)A;
            configuracion.setColor(Color.GRAY);
        }
    }
    public static class BajarPincel implements Funcion{
        @Override
        public void ejecutar(Object A, ArrayList parametros) {
            Configuracion configuracion = (Configuracion)A;
            configuracion.setColor(Color.BLACK);
        }
    }
}
```

En estos fragmentos de código encontramos los más importantes de nuestro proyecto y es que aquí creemos y decimos el comportamiento que este tendrá en la ejecución del proyecto, en caso de querer añadir más funcionalidades al proyecto es qui en donde tendremos que remitirnos y evidentemente al archivo P2.y ya que necesitamos reflejar los cambios en Parser y ParserVal que se generan al compilar el archivo de YACC, recordemos también que hacemos uso de Marcos. Todo lo anterior es la funcionalidad de Java, pero esto no se podría hacer sin nuestro archivo YACC el cual ya hemos visto, al menos la gramática, aquí solamente llamamos a las funciones creadas en nuestra máquina de pila, asignamos nuestros caracteres reservados a nuestras funciones creadas con anterioridad todo esto en nuestro yylex y finalmente añadimos funcionalidad a nuestros botones en la GUI.

Por ultimo y no mucho menos importante nuestra GUI, en donde nos encontramos con el archivo Configuración que no es más que nada el que nos permite guardar las líneas que vamos añadiendo mediante un ArrayList y para esto tendremos que usar nuestra Linea.java en donde podemos cambiar el color y definimos la dirección y el tamaño que esta tendrán, se anexa Configuracion.java ya que nos permite la visualización de las líneas que hagamos, pero, como podríamos observar las grandes maravillas que podemos hacer con nuestro proyecto si no tenemos un lienzo en donde poder verla, por lo que tenemos nuestro PanelDeDibujo.java y que ademas es qui en donde dibujamos nuestro triangulo que nos indica la dirección hacia donde se dibujara la siguiente acción que hagamos. Finalmente tenemos nuestra VentanaPrincipal.java y Logos.java los cuales no tiene tanto misterio ya que en una solo ponemos los componentes que nuestro usuario interactuará con ellos y para tener un buen formato de presentación y en el otro solo ejecutamos el programa.