

Compiladores

Martínez Coronel Brayan Yosafat

Tabla de símbolos. 3CM7

28/12/2020

Práctica 3:
Robot



Robot con tabla de símbolos

Introducción

La idea de las prácticas siguientes es hacerlas de forma secuencial y cada resultado de cada una aporta una gran ventaja para la siguiente práctica. Haremos incrementos funcionales a lo que hemos elegido en la práctica 1. En nuestro caso es el robot, y los objetivos de esta práctica es poder dibujar la trayectoria y poder guardar variables. Las preguntas clave para el desarrollo son ¿Tenemos algo que ya podamos aprovechar para adelantar el código? Y ¿Cómo funciona la tabla de símbolos?

Para la primera pregunta, claro que tenemos código, la práctica anterior fue sobre Java para interfaces y es justo lo que necesitamos, un poco de Swing para el canvas en el que dibujaremos la trayectoria del robot. Ahora, ese código en realidad no guarda variables por las reglas gramaticales, pero podemos hacer que eso cambie con un par de reglas, porque ya existe la tabla de símbolos, así como otras funciones muy útiles para las siguientes prácticas. Aunque esas no usaremos aún su potencial sino en el futuro.

Lo que tenemos que hacer es eliminar lo que no nos sirve, por ejemplo, el que pueda dibujar círculos y rectángulos no nos funciona mucho, así como el cambiar de color no nos sirve. Además, hay que mejorar algunas cosas en el diseño que se realizó, ya que puede llegar a ser muy confuso al usar formas que no se deberían de utilizar, por ejemplo, que con un simple número cambie drásticamente el comportamiento de un método, como se sabe hay mejores formas como las enumeraciones, donde queda demasiado claro qué se pretende.

También debemos de cambiar las reglas para hacer que guarde en la tabla de símbolos, mantendremos algunas partes de la estructura que tiene el proyecto, porque a largo plazo conviene mucho tener separadas muchas clases, en cambio otras dentro del archivo Y vamos a cambiarlas de lugar, son demasiado importantes como para que tengan su propio archivo.

Desarrollo

Comenzamos por eliminar todo lo relacionado con las figuras de círculos y rectángulos, lo que nos deja con una interfaz que solo la clase línea implementa, claro que, aunque es útil en otros enfoques, ahora sólo hace que exista una interfaz que nadie más va a implementar en el futuro, con fines de simplicidad podemos eliminarla y dejar claro lo que hace línea directamente.

También cambiaremos muchos nombres para que sea mucho más claro, por ejemplo, la clase Paleta ahora se llamará Botones, y el nombre de la variable será botonesSuperiores porque son los botones que están hasta arriba del resto. La clase Maquina no es necesaria para esta práctica, pero lo será para la siguiente, por lo que conservaremos una copia de lo que se realice para hacer mucho más sencillo el proceso de desarrollo de la práctica 4.

Ahora, lo que tenemos que hacer para las variables es agregar un campo a la clase Símbolo que simbolice la secuencia de una variable, esto nos va a ayudar al ser una cadena para solamente concatenar las cadenas para representar una suma de trayectorias. Luego de eso, necesitaremos un constructor que tenga entre sus argumentos una cadena, sin embargo, veremos que tendremos que usar dos cuando se implemente la máquina. Después de eso necesitamos una producción nueva. Además de otra producción que permita la suma entre secuencias.

<pre> list DRAW ON ';' list DRAW OFF ';' list VAR ';' list sec ';' </pre>	Las primeras dos de estas producciones encienden el modo de dibujo, mientras que la segunda lo apaga. Aunque se usen la producción de abajo que hace que la secuencia de la variable se dibuje en el lienzo, o bien, directamente usar una secuencia en vez de guardarla en una variable. Y, hablando de eso, la última producción es lo que hace, asigna la secuencia en la variable. La variable no puede llamarse dibujar, ni on ni off.
--	---

```
| list VAR '=' sec ';'

```

Mientras que las secuencias dependen de cuatro valores como originalmente se hacía en la práctica 1: NORTE, SUR, OESTE, ESTE. Y, ahora se suman directamente como secuencia.

La parte más relevante del código es la siguiente, en el archivo .Y:

```
list :
| list ';'
| list DRAW ON ';'
| list DRAW OFF ';'
| list VAR ';'

{ dibuja = true; System.out.println("Dibujo activado"); return 1; }
{ dibuja = false; System.out.println("Dibujo activado"); return 1; }
{
    if (dibuja) {
        Simbolo s = ((Union) $2.obj).simb;

        for (int i = 0; i < s.getSecuencia().length(); i++)
            comando(s.getSecuencia().charAt(i));
    }

    return 1;
}

| list sec ';'
{
    if (dibuja) {
        Simbolo s = ((Union) $2.obj).simb;

        for (int i = 0; i < s.getSecuencia().length(); i++)
            comando(s.getSecuencia().charAt(i));
    }

    return 1;
}

| list VAR '=' sec ';'
{
    Simbolo s = ((Union) $2.obj).simb;
    s.setSecuencia(((Union) $4.obj).simb.getSecuencia());
    return 1;
}

| list VAR '=' sec ';'
{
    Simbolo s = ((Union) $2.obj).simb;
    s.setSecuencia(((Union) $4.obj).simb.getSecuencia());
    return 1;
}

;
sec:  VAR '+' VAR ';'

| NORTE sec
{
    String s1 = ((Union) $1.obj).simb.getSecuencia();
    String s2 = ((Union) $3.obj).simb.getSecuencia();
    $$ = new ParserVal(new Union(new Simbolo(" ", (short)0, s1 + s2)));
}

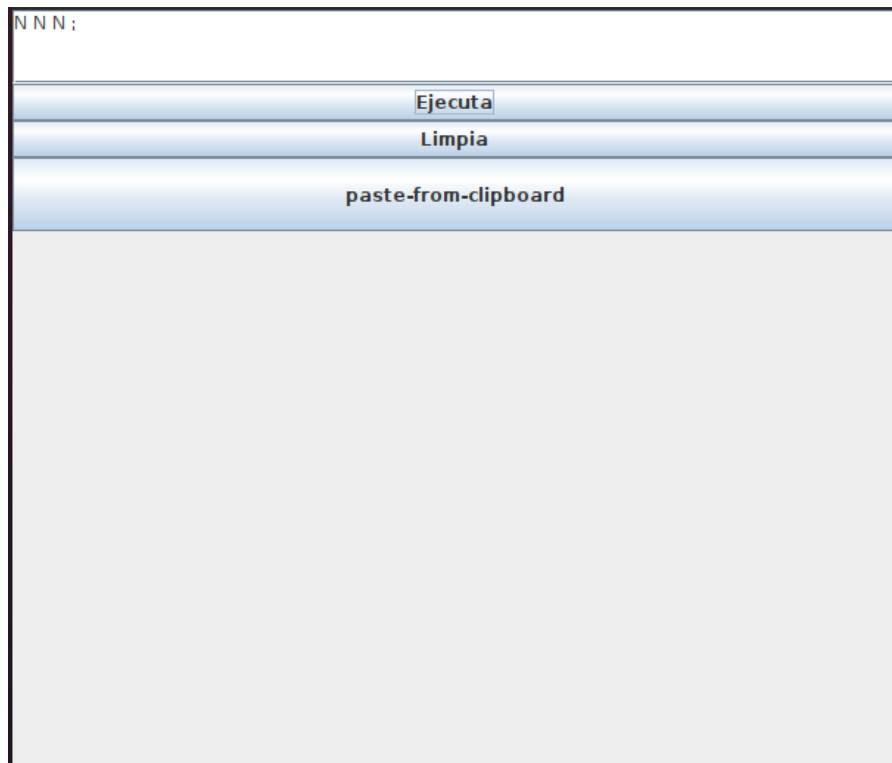
| SUR sec
{
    String sec = ((Union) $2.obj).simb.getSecuencia();
    $$ = new ParserVal(new Union(new Simbolo(" ", (short)0, "N" + sec)));
}

| ESTE sec
{
    String sec = ((Union) $2.obj).simb.getSecuencia();
    $$ = new ParserVal(new Union(new Simbolo(" ", (short)0, "S" + sec)));
}

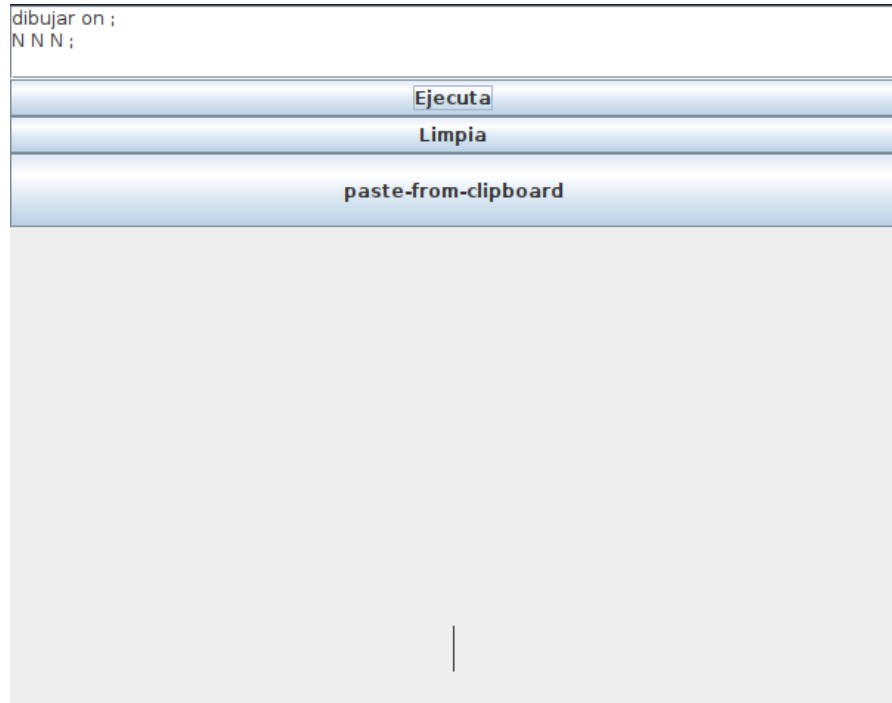
| OESTE sec
{
    String sec = ((Union) $2.obj).simb.getSecuencia();
    $$ = new ParserVal(new Union(new Simbolo(" ", (short)0, "E" + sec)));
}

{
    String sec = ((Union) $2.obj).simb.getSecuencia();
    $$ = new ParserVal(new Union(new Simbolo(" ", (short)0, "O" + sec)));
}
```

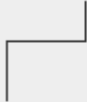
Este fragmento del código unifica las producciones de la práctica 1 con las necesarias para esta práctica. Por supuesto, también se mostrarán capturas de los resultados de usar variables y secuencias de comandos. La cosa interesante que se puede observar es que se pueden dibujar patrones más complejos con mucho menos esfuerzo que antes. Ahora los patrones repetitivos pueden ser almacenados en variables. Por supuesto, aquí se representa una suma de dos secuencias, pero, en la siguiente práctica que tiene la máquina, podremos tener una suma mucho más potente y con mucho más alcance para las secuencias.



Secuencia con el modo dibujo apagado



Secuencia con el modo dibujo prendido

ruta :
Ejecuta
Limpia
paste-from-clipboard


Usando una variable con la secuencia EEEENN

Conclusiones

Como se pudo observar, la tabla de símbolos contiene un gran poder, usar un diccionario o una lista simplemente ligada, hace que se quieran hacer muchas cosas de nivel más complejo, aunque aumentamos mucho el nivel de complejidad que podemos permitir con nuestros comandos, además de tener muchas más utilidades. Una utilidad que, aunque es buena, puede ser llevada a un nivel incluso superior. Como la función de negación, que convierte las N en S, y las O en E, y viceversa, o, la función de multiplicación, que, aunque no esté definida la multiplicación entre dos secuencias, lo está entre una secuencia y un entero, al igual que la función de suma, que no solo permitirá la suma de dos secuencias como se permite ahora, sino de múltiples secuencias. Igual, un procedimiento para imprimir el contenido de una variable que al final es una secuencia. En resumen, esto solo es un paso para crear cosas incluso más complejas, como construir una casa, un gran proyecto.