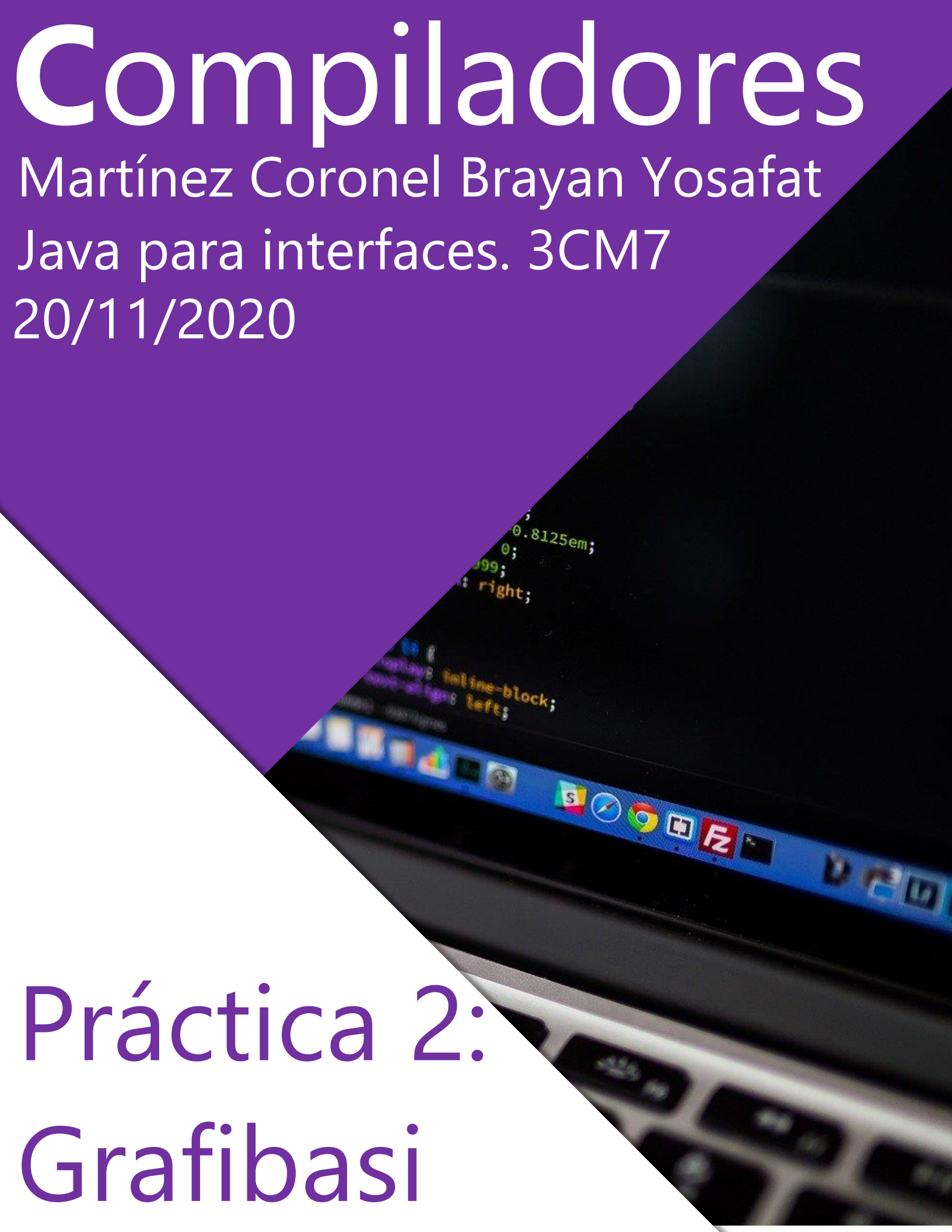
Compiladores

Martínez Coronel Brayan Yosafat

Java para interfaces. 3CM7

20/11/2020



Práctica 2: Grafibasi

Grafibasi

Introducción

Contamos con 3 carpetas con los recursos necesarios para terminar lo que se nos pide, ¿qué se nos pide?, debemos de dibujar 3 figuras, una casa, un coche y una personita. Las especificaciones las diremos más adelante, por ahora hablaremos de las carpetas.

Estas carpetas contienen código en YACC para Java, por lo que vamos a necesitar configurar y descargar el JDK y el JDE desde la terminal, además de que no podemos usar directamente el comando YACC para obtener el .java. En ese código cada versión va aumentando de funcionalidad, por ejemplo, en la primera carpeta solo dibuja círculos, rectángulos y líneas, además de cambiar de color, en la segunda le aumenta pegar directo del portapapeles y así sucesivamente.

Para nuestro caso, sólo requerimos dibujar, sin embargo, pegar directamente del portapapeles es muy útil para nosotros, así que, usaremos la segunda carpeta. Ahora bien. Los dibujos tienen algunas especificaciones, el coche, debe estar hecho de dos rectángulos y dos círculos como llantas, mientras que la casa debe tener un triángulo como techo, dos rectángulos, pero, no hay triángulos en las formas dibujables, pero sí hay líneas. Para finalizar, la persona es de palo, es decir, 3 líneas representan todo el cuerpo y la cabeza se representa por un círculo.

Como prueba, tendremos un archivo con las cadenas para dibujar cada una de las figuras que se nos piden. En esta práctica sólo mostraré los fragmentos que fueron modificados para que funcionara. Aún así, platicaremos un poco del código restante, que consiste para el dibujo de tres clases que dibujan implementando una interfaz, el resto es YACC y una máquina y tabla que almacenan las instrucciones para dibujar.

La parte de YACC tiene los tokens para cada figura y además se encarga de permitir más de una instrucción a la vez, es decir, podemos dibujar varias cosas de una sola ejecución, solo tenemos que poner punto y coma después de cada comando, la sintaxis de cada comando será platicada en la siguiente sección en donde se muestra la captura de pantalla de las cadenas para dibujar las 3 figuras.

Desarrollo

Ahora, ya que tenemos claro qué se requiere en la práctica, pasemos directamente al archivo que contiene las cadenas:

```
1 //Coche
2 rectangulo 50 50 100 20 ;
3 rectangulo 60 30 70 20 ;
4 circulo 10 60 70 ;
5 circulo 10 120 70 ;
6
7 //Casa
8 rectangulo 50 200 100 90 ;
9 rectangulo 110 250 30 40 ;
10 line 50 200 100 150 ;
11 line 100 150 150 200 ;
12
13 //Personita
14 circulo 50 250 250 ;
15 line 275 300 275 400 ;
16 line 250 330 300  330 ;
17 line 245 430 275 400 ;
18 line 275 400 305 430 ;
```

Cada instrucción está dada por el nombre de lo que se quiere dibujar y después los parámetros necesarios separados por espacios, en este caso todos son dobles, pero un dato importante es que a la hora de dibujar son cambiados a enteros en la implementación de la interfaz dibujable.

Las partes que se modificaron fueron en Maquina.java y en el archivo .y en el primero hace más pop en la pila y en el segundo hace más push de constantes, ya que inicialmente sólo dibujaba las figuras en la misma zona:

```
void line() {
        double d4 = ((Double)pila.pop()).doubleValue();
        double d3 = ((Double)pila.pop()).doubleValue();
        double d2 = ((Double)pila.pop()).doubleValue();
        double d1 = ((Double)pila.pop()).doubleValue();

        if (g! = null) (new Linea((int)d1,(int)d2,(int)(d3),(int)(d4))).dibuja(g);
}

void circulo() {
        double d3 = ((Double)pila.pop()).doubleValue();
        double d2 = ((Double)pila.pop()).doubleValue();
        double d1 = ((Double)pila.pop()).doubleValue();

        if (g! = null) (new Circulo((int)d2, (int)d3, (int)d1)).dibuja(g);
}

void rectangulo() {
        double d4=((Double)pila.pop()).doubleValue();
        double d3=((Double)pila.pop()).doubleValue();
        double d2=((Double)pila.pop()).doubleValue();
        double d1=((Double)pila.pop()).doubleValue();

        if(g!=null) (new Rectangulo((int)d1, ((int)d2), (int)d3, ((int)d4) )).dibuja(g);
}
```

Archivo Maquina.java

```
| 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);
                                          maq.code("constpush");
                                          maq.code(((Algo)$3.obj).simb);
                                          maq.code("constpush");
                                          maq.code(((Algo)$4.obj).simb);
                                          maq.code("circulo");
                                        }
```

Archivo forma.y

Con estos cambios ahora mediante el canvas que contiene el JFrame se dibujan las 3 formas de manera libre, por supuesto limitado al tamaño del canvas, ahora cada producción inserta las constantes necesarias para cada forma, por ejemplo, ahora se insertan 3 constantes cuando es un círculo, inserta 4 cuando es un rectángulo o una línea, cada comando está dado como:

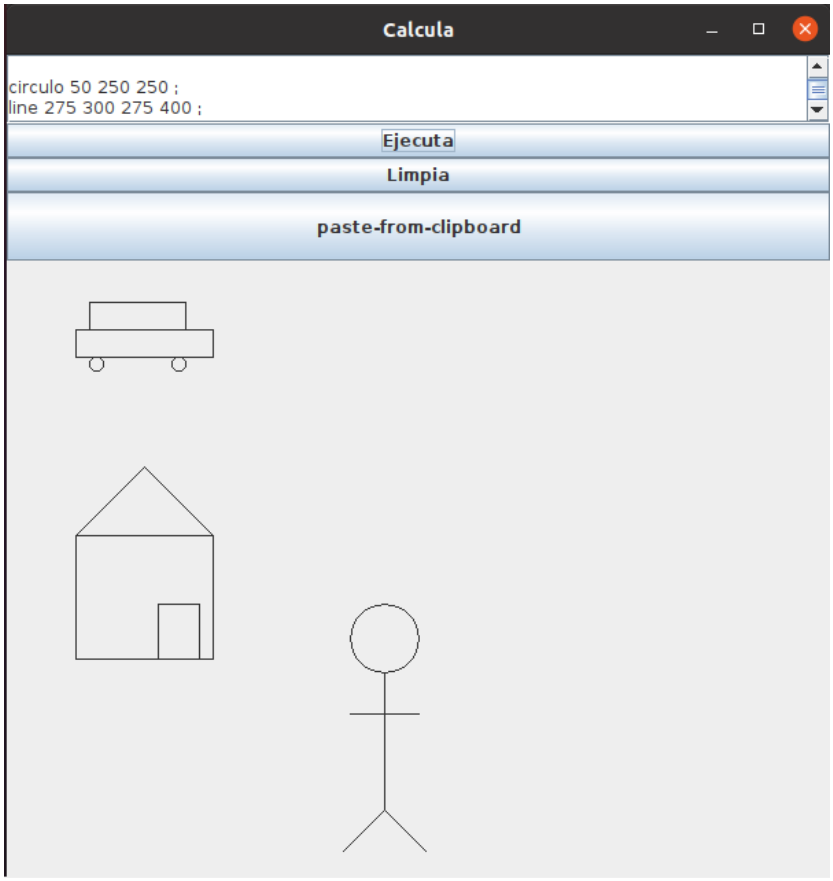
Línea: line x1 y1 x2 y2 ;

Círculo: circulo r x y ;

Rectángulo: rectángulo x1 y1 x2 y2 ;

Donde las últimas letras representan números, el espacio entre ellas es importante y también lo es el espacio que existe entre el punto y coma y el último número entero.

Ingresando las líneas del archivo que mostramos al inicio, se obtienen las siguientes figuras:



Conclusiones

En la documentación de byacc para Java el chico que lo realizó dijo que lo hizo porque una persona dijo que no se podía tener YACC para Java, justo antes de decir que quien lo dijo era una persona tonta. Me parece sumamente útil, porque personalmente me gusta mucho Java, porque tiene una escritura muy estricta en el sentido de que no te permite realizar tantas prácticas malas como en otros lenguajes. Creo que es muy útil porque como muchos, yo me apego mucho al paradigma orientado a objetos, así que tener esta oportunidad lo hace mucho más sencillo, ya que escribir código, diseñar su estructura me parece un arte en Java, es pasar muy directo de un diseño UML a código.