

Símbolos

Superclases, subclases y extends

Cátedra 14

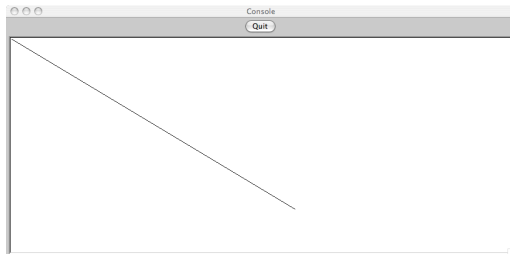
Otoño 2005

La clase de hoy es un taller donde se muestra el uso de lo aprendido en la última clase. El problema consiste en escribir un programa que permita dibujar expresiones complejas en la pantalla.

1 drawLine

La clase Console tiene un método que permite dibujar líneas llamado drawLine. Este método recibe 2 pares de coordenadas:

```
public class Ejx {
    public static void main (String args[]) {
        Console c=new Console();
        c.drawLine(0,0,500,300);
    }
}
```



2 La clase Letra

La expresión más sencilla que podemos dibujar es una letra. Por simplicidad, supondremos que el alfabeto tiene solo 3 letras: o, + y x.

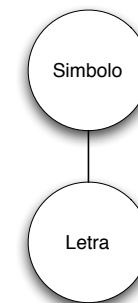
```
class Letra extends Simbolo {
    char l;
    public Letra(char c) {
        this.l=c;
    }
}
```

```
public void dibuja(Console c, int x1,int y1, int x2, int y2) {
    if (l=='o')
        c.drawOval(x1,y1,x2-x1,y2-y1);
    else if (l=='x') {
        c.drawLine(x1,y1,x2,y2);
        c.drawLine(x1,y2,x2,y1);
    } else if (l=='+') {
        int xm=x1/2+x2/2;
        int ym=y1/2+y2/2;
        c.drawLine(xm,y1,xm,y2);
        c.drawLine(x1,ym,x2,ym);
    }
}
```

La clase Símbolo es

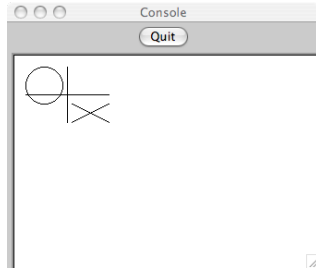
```
class Simbolo {
    public void dibuja(Console c, int x1,int y1, int x2, int y2) {
    }
}
```

Es decir, tenemos la siguiente jerarquía



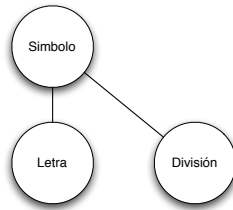
El siguiente código, al ejecutarse dibuja la figura inferior:

```
public class Ejemplo {
    public static void main (String args[]) {
        Console c=new Console();
        Letra l=new Letra('o');
        Letra m=new Letra('x');
        Letra n=new Letra('+');
        l.dibuja(c,10,10,50,50);
        m.dibuja(c,60,50,100,70);
        n.dibuja(c,10,10,100,70);
    }
}
```



3 La clase división

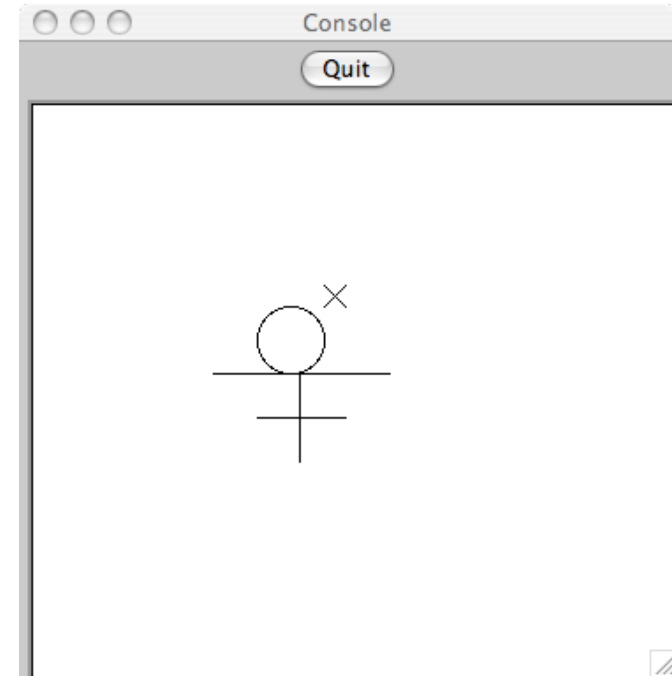
Vamos a crear una nueva clase en la Jerarquía: la división:



```
class Division extends Simbolo{
    Simbolo l1,l2;
    public Division(Simbolo l, Simbolo m) {
        this.l1=l;
        this.l2=m;
    }
    public void dibuja(Console c, int x1,int y1, int x2, int y2) {
        int h=x2-x1;
        int v=y2-y1;
        this.l1.dibuja(c,x1+h/4,y1,x2-h/4,y1+v/2);
        this.l2.dibuja(c,x1+h/4,y1+v/2,x2-h/4,y2);
        c.drawLine(x1,y1+v/2,x2,y1+v/2);
    }
}
```

4 De qué sirvió todo

Podemos dibujar la división donde el numerador es una exponencial...



```
public class Ejemplo2 {
    public static void main (String args[]) {
        Console c=new Console();
        Letra l=new Letra('o');
        Letra m=new Letra('x');
        Exponente e=new Exponente(l,m);
        Division d=new Division(e,new Letra('+'));
        d.dibuja(c,100,100,200,200);
    }
}
```

AL MARGEN

Este patrón se llama **Composición**, pues un objeto está compuesto de objetos de la misma jerarquía. Permite crear sistemas complejos a partir de objetos sencillos. Aplicaciones: mapas, circuitos, diagramación de ecuaciones.