

# Panel de Dibujo

*Jorge A. López Vargas*

Este documento tiene como finalidad el explicar algunas de las características que posee Java para la construcción de aplicaciones con interfaz gráfica de usuario.

La aplicación final tendrá la siguiente apariencia (para realizar dibujos debe presionar el botón izquierdo del ratón y arrastrarlo).



La aplicación consta de 2 clases:

FrmPrincipal: El JFrame principal que equivale a la ventana.

PnlDibujo: Un objeto del tipo JPanel que permite realizar los dibujos.

Analicemos a detalle la clase PnlDibujo que es la clase que realiza todo el trabajo.

```
15 public class PnlDibujo extends javax.swing.JPanel {
16
17     private int x;
18     private int y;
19
20     /** Creates new form PnlDibujo */
21     public PnlDibujo() {
22         initComponents();
23         x = -1;
24         y = -1;
25     }
```

Decimos que la clase es del tipo JPanel porque vemos como hereda de dicha clase (vea la definición de la clase). Tiene 2 atributos o propiedades x e y, que representan las coordenadas de la posición del puntero del ratón. Tenemos un constructor que llama a un método que es creado por NetBeans y que crea los componentes GUI, que en éste caso no son existen; también se inicializa los atributos a -1.

Este panel, debe reaccionar a un evento del ratón, y para ello necesita de un escuchador de dicho evento, que se define en el método initComponents(). Ponga atención a las líneas desde la 44 a la

48. El evento al que se debe reaccionar es el arrastrar (presionar el botón izquierdo y arrastrar el ratón). Cuando se produce el evento arrastrar se ejecutará el método `formMouseDragged`.

```
42 private void initComponents() {
43
44     addMouseListener(new java.awt.event.MouseAdapter() {
45         public void mouseDragged(java.awt.event.MouseEvent evt) {
46             formMouseDragged(evt);
47         }
48     });
49 }
```

El evento `formMouseDragged` básicamente lo que hace es obtener las coordenadas `x` e `y` del puntero del mouse y asignarlas a los atributos correspondientes y llamar a un evento llamado `updateUI` que es heredado de la clase padre – `JPanel` y que básicamente llama al método `repaint` (que analizaremos más adelante).

```
62 private void formMouseDragged(java.awt.event.MouseEvent evt) {
63     x = evt.getX();
64     y = evt.getY();
65     updateUI();
66 }
```

El método `repaint` es el encargado de dibujar elementos en un contenedor. Recibe un parámetro del tipo `Graphics` que posee muchos métodos para realizar varios diferentes tipos de dibujos. Lo primero que hace el método es verificar que los valores de los atributos no sean negativos para luego configurar el color con que se dibujará y lo segundo es dibujar un óvalo relleno, los primeros parámetros son las posiciones `x` e `y` en donde se dibujará el óvalo y los últimos parámetros son el ancho y alto.

```
28 @Override
29 public void paint(Graphics g) {
30     if (x >= 0 && y >= 0) {
31         g.setColor(Color.BLUE);
32         g.fillOval(x, y, 5, 5);
33     }
34 }
```

`@Override` se conoce como anotación y le dice al compilador de Java que el método `paint` ha sido sobrescrito en la clase `PnlDibujo` que es hija de la clase `JPanel`.

Finalmente debemos agregar este panel al frame principal y lo hacemos en el constructor de la clase `FrmPrincipal`. Para ello se debe crear un objeto del tipo `PnlDibujo`, luego fijar el tamaño con el método `setSize`, el tamaño debe ser el mismo de la ventana para ello se usa la sentencia `this.getSize()` para obtener el tamaño actual de la ventana. Finalmente se agrega el objeto `pnlDibujo` al `FrmPrincipal`.

```
13 public class FrmPrincipal extends javax.swing.JFrame {
14
15     /** Creates new form FrmPrincipal */
16     public FrmPrincipal() {
17         initComponents();
18         PnlDibujo pnlDibujo = new PnlDibujo();
19         pnlDibujo.setSize(this.getSize());
20         add(pnlDibujo);
21     }
```