| **Objetivos** | Desarrollar una interfaz gráfica de usuario. Trabajar con clases de AWT. Trabajar con diferentes gestores de organización. Realizar clases para el tratamiento de eventos. |
|---|---|

# Índice

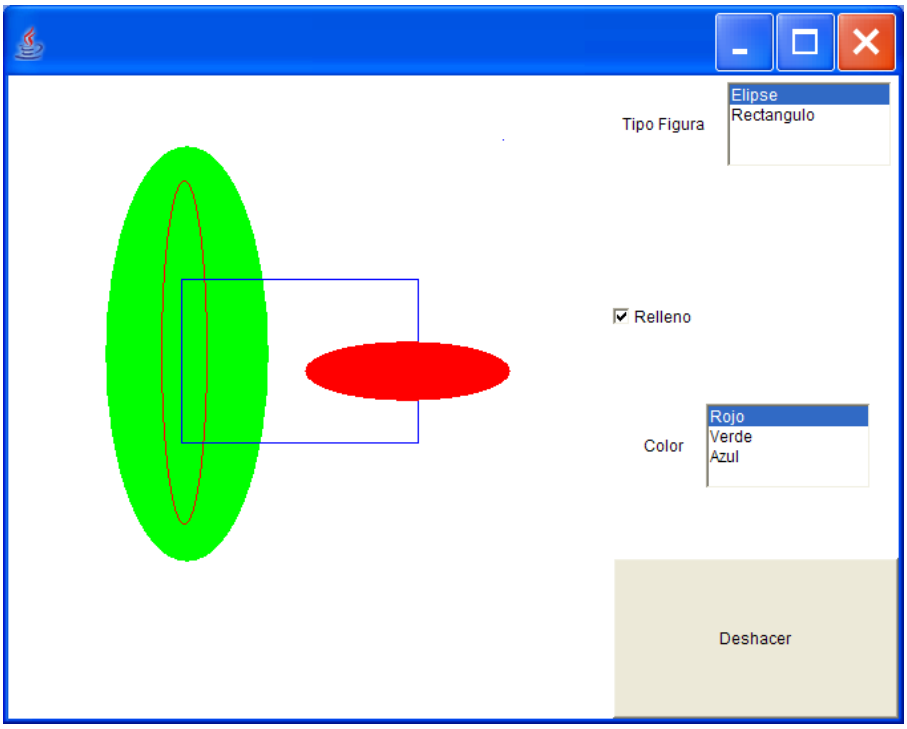


---

## 1. Aplicación de dibujo

En esta práctica se pide realizar una interfaz gráfica de usuario que permita dibujar diferentes figuras.

La aplicación se muestra en la siguiente figura:



Como se puede ver, se debe permitir al usuario seleccionar entre una serie de figuras (una elipse o un rectángulo), seleccionar si la figura debe ir rellena o no, se debe permitir seleccionar el color con el que se va a pintar la siguiente figura y se debe permitir deshacer (es decir, borrar) de forma consecutiva las figuras que se han ido añadiendo.

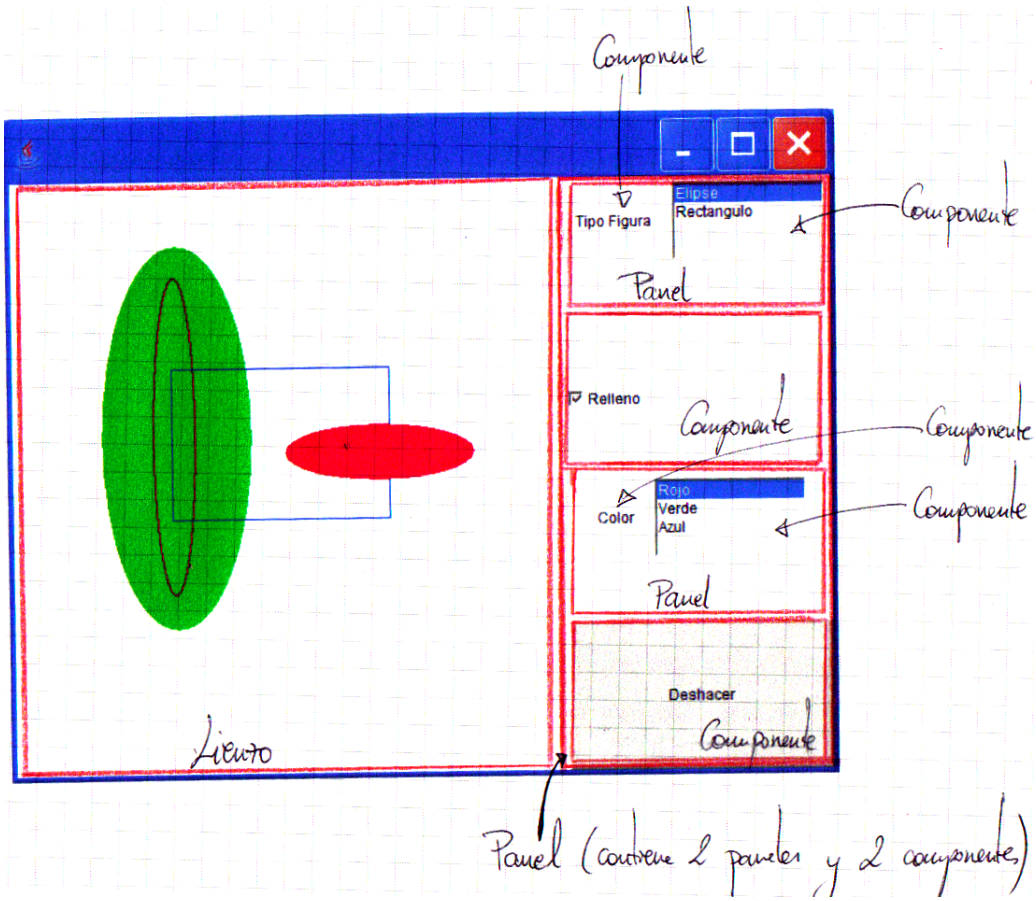
La información que va seleccionando el usuario se puede almacenar en un objeto del tipo `HashMap` (igual que se hacía en el ejemplo `EventosItem.java` mostrado en teoría).

Una vez que el usuario ha seleccionado la información sobre la figura, puede pulsar en el área de la derecha y mover el ratón hasta el punto final (pulsa en el punto inicial y suelta en el punto final), con lo que se dispondrá de toda la información necesaria para crear y mostrar la figura (tipo de figura, relleno, color y coordenadas).

Las figuras que se van creando se deben almacenar en un objeto del tipo `Vector`.

## 1.1. Organización de contenedores y componentes

La siguiente figura muestra una posible forma de organizar los componentes y los contenedores.



Hay que decidir qué gestores de organización se van a utilizar tanto en la ventana como en el panel que contiene las opciones.

## 2. Clases a realizar

Se debe realizar una *interface* y una serie de clases que se detallan a continuación.

La *interface* indica los métodos que deben tener las figuras a utilizar en la aplicación (en este caso sólo un método):

```
1
2  public interface Figura {
3     public void pinta(Graphics g);
4  }
```

Se deben realizar dos clases que implementen a esta *interface* (en rojo se muestra lo que hay que completar):

```
1
2  // import
3
4  public class Elipse implements Figura {
5     // Atributos
6
7     // Constructor
8     Elipse(int x1,int y1,int x2,int y2,Color c, boolean fill){...}
9
10    // El metodo que define la interface
11    public void pinta(Graphics g) {...}
12
13 }
```

```
1
2  // import
3
4  public class Rectangulo implements Figura {
5     // Atributos
6
7     // Constructor
8     public Rectangulo(int x1,int y1,int x2, int y2,Color c, boolean fill){...}
9
10
11    // El metodo que define la interface
12    public void pinta(Graphics g) {...}
13
14 }
```

Finalmente se debe realizar la clase `Dibujo` que extienda a la clase `Frame` y una serie de clases internas que servirán para tratar los eventos (aunque estas clases también podrían ser externas a la clase `Dibujo`). El esqueleto de esta clase sería:

```
1
2  // import
3
4  public class Dibujo extends Frame {
5     // Para almacenar las diferentes figuras que se van mostrando
6     private Vector figs = new Vector();
7
```

```
8     // List para ofrecer diferentes figuras
9     private List listaFiguras;
10
11    // Checkbox para seleccionar figura rellena o no
12    private Checkbox  relleno;
13
14    // List para ofrecer diferentes colores
15    private List listaColores;
16
17    // Boton para deshacer (de forma indefinida)
18    private Button deshacer;
19
20    // HashMap para almacenar la informacion seleccionada por el usuario
21    private HashMap informacion = new HashMap();
22
23    // Lienzo es un Panel donde dibujar
24    private Lienzo lienzo;
25
26
27    // Clase Lienzo donde se mostraran las figuras
28    class Lienzo extends Panel{
29       // Ocultamos paint(Graphics g) de la clase Panel
30       public void paint(Graphics g){
31
32          // Si hay figuras en el Vector lo recorremos y las vamos pintando
33          if (figs.size()>0)
34             for (int i=0; i<figs.size(); i++)
35                   ((Figura)figs.get(i)).pinta(g);
36       }
37
38       // Metodo que crea una nueva figura a partir de la información que
39       // contiene el HashMap y de las coordenadas que se pasan,
40       // almacena la figura en el Vector y repinta el Lienzo
41       public void dibuja(int x1,int y1,int x2,int y2) {...}
42    }
43
44    // Clase para tratar los eventos del tipo ItemEvent
45    class TrataEventosItem implements ItemListener {...}
46
47    // Clase para tratar los eventos del tipo MouseEvent sobre el boton
48    class TrataEventosRatonBoton extends MouseAdapter {...}
49
50    // Clase para tratar los eventos del tipo MouseEvent sobre el Lienzo
51    // Se deben implementar mousePressed y mouseReleased
52    class TrataEventosRatonBoton extends MouseAdapter {
53       // Atributos
54
55       public void mousePressed(MouseEvent e){...}
56
57       public void mouseReleased(MouseEvent e){...}
58
59    }
60
61    // Constructor
62    public Dibujo() {
63
64       informacion.put("TipoFigura","Elipse");
65       informacion.put("Color","Rojo");
66       informacion.put("Relleno","NO");
67
68       // Creacion del lienzo y de los paneles con los componentes
69
70       // Registro de auditores de eventos
71
72       // Añadimos a la ventana el Lienzo y el panel de opciones
73
74       setSize(800,500);
75       show();
76
77    }
```

```
78
79      public static void main(String[] args) {
80          new Dibujo();
81      }
82  }
```

**Nota**

En el tema 2 a propósito de la diapositiva 38, en la que se comenta que en la programación procedural el interés se centra en la descomposición en subrutinas (funcionalidad) y que es inestable ya que depende de los cambios en los requisitos funcionales, se puso un ejemplo en la pizarra de programa procedural que permitía dibujar figuras. Se puso de manifiesto las dificultades que surgían si se deseaba añadir una nueva figura o si se deseaba permitir deshacer los cambios. Como se puede observar, con la versión orientada a objetos, añadir una nueva figura es sencillo (hay que modificar pocas líneas de código simplemente crear una nueva clase y añadir una opción a la lista de figuras). Además la opción deshacer también resulta trivial.