Interfaces gráficas de usuario (GUI)

Swing

Tipos de programas en Java

- Aplicaciones
  - Se pueden ejecutar directamente en un entorno Java
  - Tipos
    - Modo de consola
      - ***Interacción mediante teclado***
      - ***Interfaz basado en texto***
    - Aplicaciones con interfaz gráfico (GUI)
      - ***Ventanas graficas para entrada y salida de datos***
      - ***Iconos***
      - ***Dispositivos de entrada (e.g. ratón, teclado)***
      - ***Interacción directa***
- Applets
  - Pequeñas aplicaciones que se ejecutan dentro de un navegador (o en el visualizador de applets - *Appletviewer*)
    - Interfaz gráfico
    - Limitaciones por motivos de seguridad

Bibliotecas de componentes para GUI

- Abstract Windowing Toolkit (AWT)
  - "Look & Feel" dependiente de la plataforma
    - ***La apariencia de ventanas, menúes, etc. es distinta en Windows, Mac, Motif, y otros sistemas***
  - Funcionalidad independiente de la plataforma
  - Básico y experimental
  - Estándar hasta la versión JDK 1.1.5
- Swing / Java Foundation Classes (desde JDK 1.1.5)
  - "Look & Feel" y funcionalidad independiente de la plataforma ("Java Look & Feel")
    - ***Los menús y controles son como los de las aplicaciones "nativas"***
    - ***A las aplicaciones se les puede dar una apariencia en función de la plataforma específica***
  - Nuevas funcionalidades
    - ***API de accesibilidad para personas con necesidades específicas***

Elementos básicos

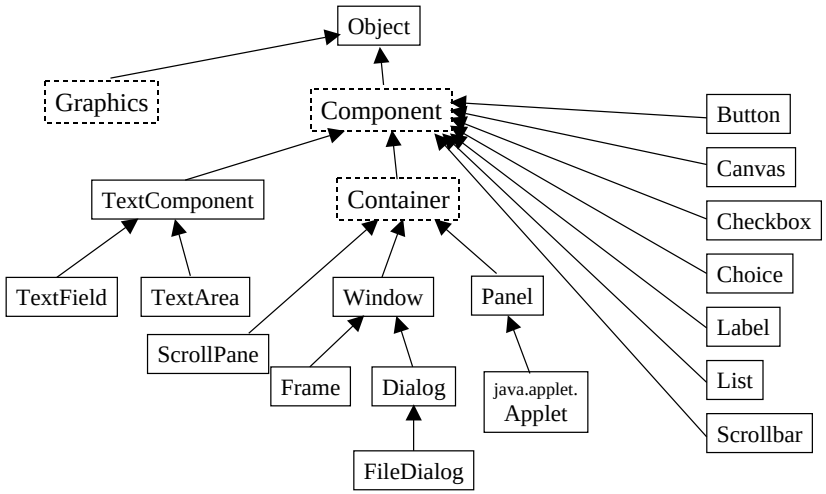
- Componentes GUI (widgets)
  - Objetos visuales del interfaz
  - Un programa gráfico es un conjunto de componentes anidados
    - ventanas, contenedores, menús, barras, botones, campos de texto, etc.
- Administradores de diseño o disposición (*layout managers*)
  - Gestionan la organización de los componentes gráficos de la interfaz
- Creación de gráficos y texto - Clase *Graphics*
  - Define fuentes, pinta textos,
  - Para dibujo de líneas, figuras, coloreado,...
- Interactividad: manejo de eventos
  - Ratón
  - Teclado

Componentes del Swing

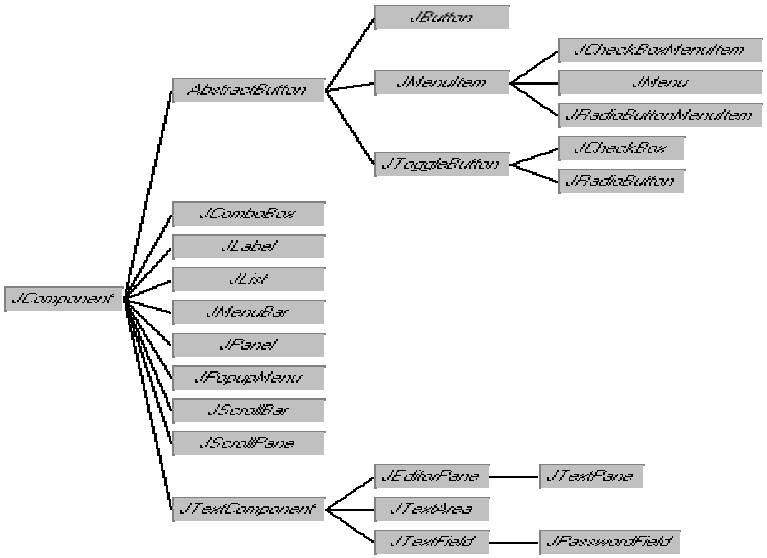
- Contenedores
  - Contienen otros componentes (o contenedores)
    - Estos componentes se tienen que añadir al contenedor y para ciertas operaciones se pueden tratar como un todo
    - Mediante un gestor de diseño controlan la disposición (*layout*) de estos componentes en la pantalla
    - Ejemplo: JPanel, JFrame, JApplet
- Lienzo (clase *Canvas*)
  - Superficie simple de dibujo
- Componentes de interfaz de usuario
  - botones, listas, menús, casillas de verificación, campos de texto, etc.
- Componentes de construcción de ventanas
  - ventanas, marcos, barras de menús, cuadros de diálogo

Jerarquía de componentes del AWT

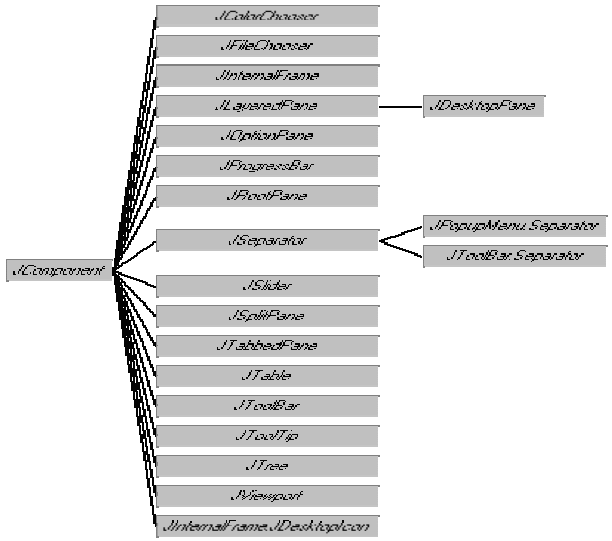
Jerarquía de clases



Jerarquía de componentes de Swing (I)



Jerarquía de componentes de Swing (II)

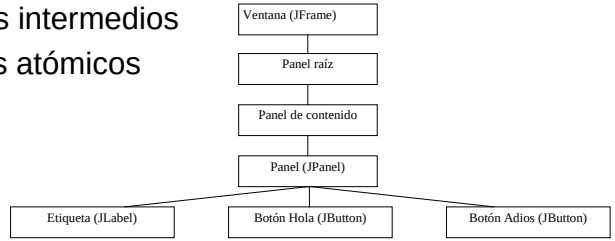


Ejemplo - GUI simple con un *JFrame*

```
import javax.swing.*;
public class GUISimple extends JFrame {
        public GUISimple (){
                setSize(200, 100);
                setVisible(true);
        }
        public static void main(String args[]) {
                GUISimple ventana = new GUISimple();
                ventana.setTitle("ventana tipo frame");
        }
}
```

Jerarquía de composición

- Contenedores de alto nivel
- Contenedores intermedios
- Componentes atómicos

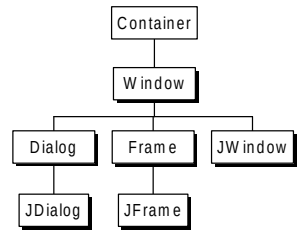


Clases básicas

- java.awt.Component
  - Esta clase abstracta define la funcionalidad básica de todos los componentes gráficos en Java
- java.awt.Container
  - Clase abstracta que permite agrupar uno o varios componentes de forma que se puedan tratar como una unidad.
  - Proporciona métodos para añadir y eliminar componentes o para definir el tipo de presentación que se realiza de los componentes en la pantalla (mediante layout Managers)
- javax.swing.JComponent
  - Es la clase base de casi todos los componentes de interacción que incorpora Swing excepto los contenedores de alto nivel (p.e. *JFrame*).

Contenedores de alto nivel

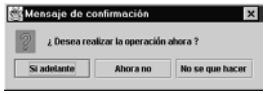
- javax.swing.JFrame
  - Habitualmente la clase *JFrame* se emplea para crear la ventana principal de una aplicación en Swing
- javax.swing.JDialog
  - Genera ventanas secundarias de interacción con el usuario
    - Cuadros de diálogo
  - Son modales: el usuario no puede interactuar con otra ventana hasta que no cierre la actual



Cuadros de diálogo: JOptionPane

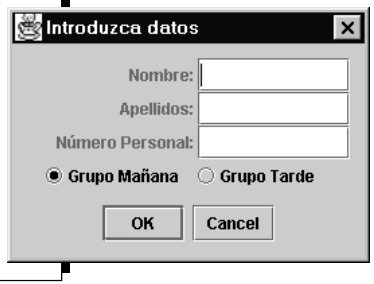
```
import javax.swing.*;
//icono y boton OK predeterminados, se pone el título
JOptionPane.showMessageDialog(ventana,
     "Debe introducir datos en todos los campos", // mensaje
     "Error de entrada de datos", // título
     JOptionPane.ERROR_MESSAGE); // icono

// cuadro de opción personalizado
Object[] textoOpciones = {"Si adelante", "Ahora no",
                                  "No se que hacer"};
int opcion = JOptionPane.showOptionDialog(ventana,
            "¿ Desea realizar la operación ahora ?",
            "Mensaje de confirmación",
            JOptionPane.YES_NO_CANCEL_OPTION,
            JOptionPane.QUESTION_MESSAGE,
            null,     //utilizar el icono predeterminado
            textoOpciones,
            textoOpciones[0]); //botón predeterminado
}
```



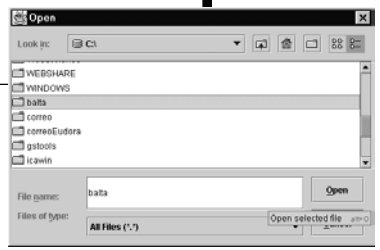
Petición de datos con JOptionPane

```
import javax.swing.*;
if (JOptionPane.showConfirmDialog(this
      ,panel  // panel que se muestra dentro del cuadro de diálogo
      ,"Introduzca datos"
      ,JOptionPane.OK_CANCEL_OPTION
      ,JOptionPane.PLAIN_MESSAGE
     ) == JOptionPane.OK_OPTION)
{
      String nombre = campoNombre.getText();
      String apellidos = campoApellidos.getText();
      int numPer=0;
      try {
          numPer= Integer.parseInt(campoNP.getText());
          errorNumero = false;
       } catch (NumberFormatException nfe) {
                errorNumero = true;
                mostrarCuadroError(.....);
}
```



JFileChooser

```
import javax.swing.*;
// se crea el selector de ficheros
JFileChooser selector = new JFileChooser();
// solo posibilidad de seleccionar directorios
selector.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
// se muestra; se comprueba si el usuario acepta o cancela
int opcion = selector.showOpenDialog(null);
if (opcion == JFileChooser.APPROVE_OPTION) {
   //obtenemos el fichero o directorio seleccionado
   File archivo = selector.getSelectedFile();
   System.out.println("archivo seleccionado: " + archivo);
}
else
 System.out.println("operacion cancelada ");
```



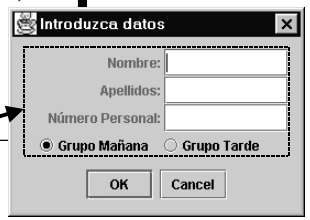
Contenedores intermedios

- JPanel
  - Agrupa a otros componentes
  - No tiene presentación gráfica pero se le pueden añadir bordes o cambiar el color de fondo
- JScrollPane
  - Incluye barras de desplazamiento

Panel con datos del usuario

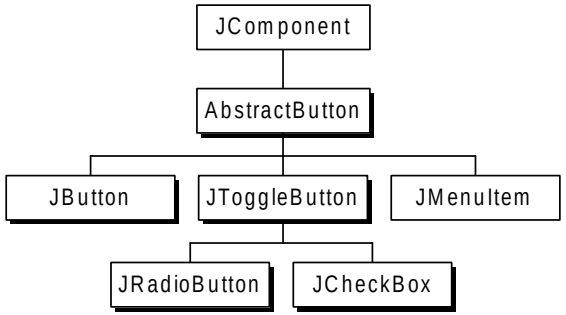
```
JPanel panel = new JPanel(new GridLayout(4,2));
JLabel etiquetaNombre = new JLabel("Nombre: ", JLabel.RIGHT);
JTextField campoNombre = new JTextField();
panel.add(etiquetaNombre);
panel.add(campoNombre);
JLabel etiquetaApellidos = new JLabel("Apellidos: ", JLabel.RIGHT);
JTextField campoApellidos = new JTextField();
panel.add(etiquetaApellidos);
panel.add(campoApellidos);
JLabel etiquetaNP = new JLabel("Número Personal: ", JLabel.RIGHT);
JTextField campoNP = new JTextField();
panel.add(etiquetaNP);
panel.add(campoNP);
ButtonGroup grupoBotones = new ButtonGroup();
JRadioButton mañana = new JRadioButton("Grupo Mañana", true);
JRadioButton tarde = new JRadioButton("Grupo Tarde");
grupoBotones.add(maÒana);
grupoBotones.add(tarde);
panel.add(maÒana);
panel.add(tarde);
```

Panel que agrupa, tres etiquetas, tres campos de texto y dos botones de radio

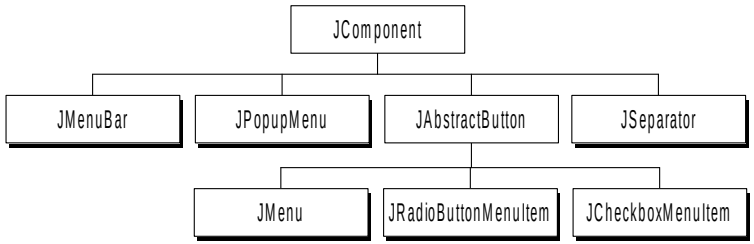


Iconos, Etiquetas y Botones

- Iconos
  - Elementos gráficos que se pueden añadir a los componentes
- Etiquetas
  - Elementos para mostrar información
- Botones

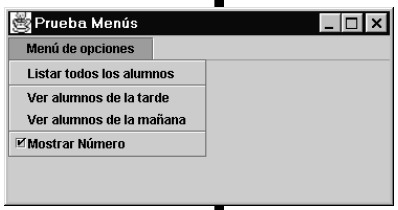


Menús

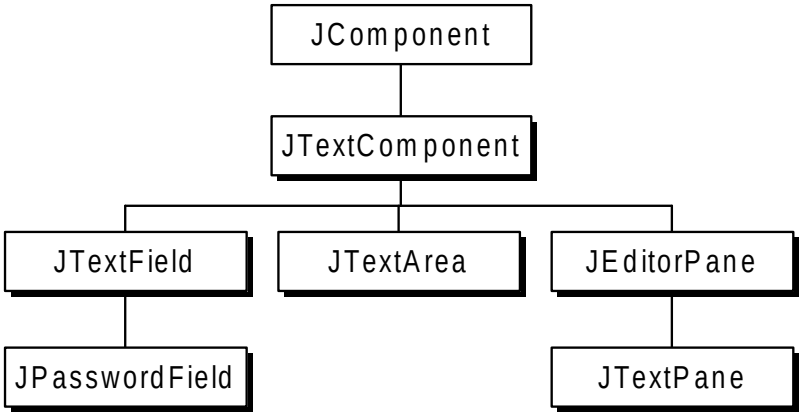


Ejemplo menús

```
import javax.swing.*;
JMenuBar barraMenu = new JMenuBar();
JMenu menuOpciones = new JMenu("Menú de opciones");
JMenuItem listar = new JMenuItem("Listar todos los alumnos");
menuOpciones.add(listar);
// separador
menuOpciones.add(new JSeparator());
JMenuItem listarTarde = new JMenuItem("Ver alumnos de la tarde");
menuOpciones.add(listarTarde);
JMenuItem listarMañana = new JMenuItem("Ver alumnos de la
mañana");
menuOpciones.add(listarMañana);
menuOpciones.add(new JSeparator());
JCheckBoxMenuItem verNumero = new
JCheckBoxMenuItem("Mostrar Número");
menuOpciones.add(verNumero);
barraMenu.add(menuOpciones);
// establecer como barra de menús
// en contenedor de alto nivel
setJMenuBar(barraMenu);
```



Elementos de manejo de texto

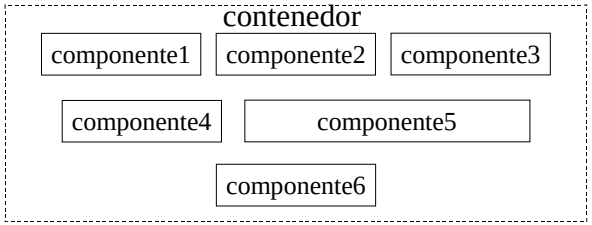


Administrador de diseño (*layout manager*)

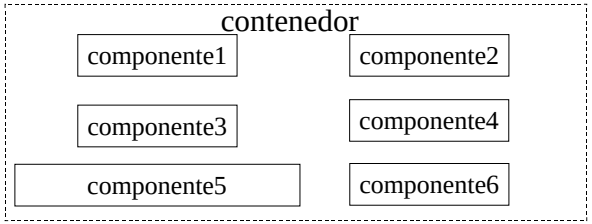
- Cómo se colocan los componentes (usando el método *add*) depende de la composición (*layout*)
- Tipos de diseños o composiciones
  - FlowLayout
    - Los componentes se ponen de izquierda a derecha hasta llenar la línea, y se pasa a la siguiente. Cada línea se centra
      - ***Por defecto, en paneles y applets***
  - BorderLayout
    - Se ponen los componentes en un lateral o en el centro
    - se indica con una dirección:"East", "West", "North", "South", "Center"
      - ***Por defecto, en marcos***
  - GridLayout
    - Se colocan los componentes en una rejilla rectangular (filas x cols)
    - Se añaden en orden izquierda-derecha y arriba-abajo
- Para poner un layout se utiliza el método *setLayout():*

```
GridLayout  nuevolayout = new GridLayout(3,2);
setLayout(nuevolayout);
```


Administrador de diseño



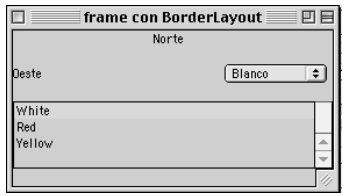
FlowLayout



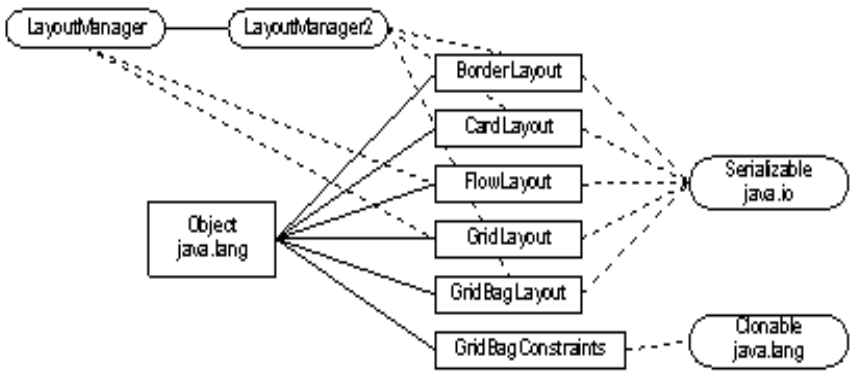
GridLayout(3,2)

Otros administradores

- GridBagLayout
  - Similar al GridLayout pero mas versátil
  - Presenta los componentes en una rejilla, pero:
    - Un componente puede ocupar más de una fila y más de una columna
    - Las filas y las columnas pueden tener tamaños diferentes
    - No se tiene que rellenar en un orden predeterminado
  - Utiliza *GridBagConstraints* para especificar como deben colocarse, distribuirse, alinearse, etc., los componentes



Administradores de diseño



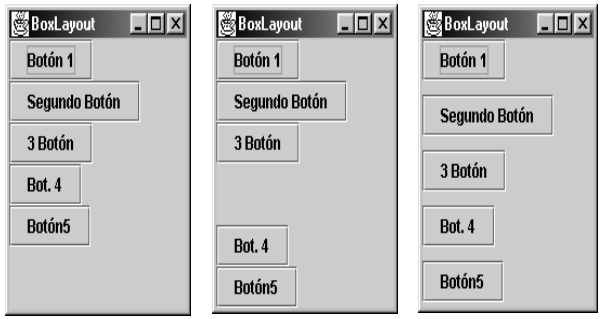
Nuevos administradores de diseño en Swing

- *BoxLayout*
  - Organiza los componentes en una única fila o columna
    - Por defecto el espacio libre se deja al final
  - Los elementos pueden tener distinto tamaño y alineación
- Normalmente se utiliza conjuntamente con la clase *Box*
  - Permite crear componentes invisibles que ocupan un tamaño fijo para mejorar la presentación (áreas rígidas y *struts*)
  - Permite crear "gomas extensibles" o componentes invisibles que también se redimensionan cuando se redimensiona el contenedor

Ejemplo BoxLayout

```
public class PruebaBoxLayout extends JFrame {
   PruebaBoxLayout(){
        JButton b1, b2, b3, b4, b5;
        b1 = new JButton("Botón 1");b2 = new JButton("Segundo Botón");
        b3 = new JButton("3 Botón");b4 = new JButton("Bot. 4");
        b5 = new JButton("Botón5");
        JPanel panel = new JPanel();
        // se asigna un BoxLayout vertical al panel
        panel.setLayout( new BoxLayout(panel, BoxLayout.Y_AXIS));
        // se añaden los botones al panel con glue entre ellos
        panel.add(b1); panel.add(Box.createGlue());
        panel.add(b2); panel.add(Box.createGlue());
        panel.add(b3); panel.add(Box.createGlue());
        panel.add(b4); panel.add(Box.createGlue());
        panel.add(b5);
        getContentPane().add(panel);
        setTitle("BoxLayout");
        pack(); setVisible(true);
    }
   public static void main(String args[]) {
        PruebaBoxLayout ventana = new PruebaBoxLayout();}}
```


Resultado BoxLayout



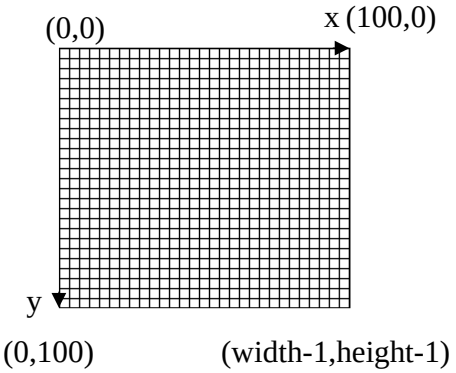
La captura de la izquierda es la situación por defecto, en la central se introduce "pegamento" entre los botones tres y cuatro, y la captura de la derecha es con "pegamento" entre todos los botones.

La clase *Graphics*

- Clase abstracta que es la base para los contextos gráficos que permiten a una aplicación dibujar los componentes independientemente del dispositivo de salida
- Un contexto gráfico es un objeto que funciona junto con las ventanas para mostrar los objetos gráficos
- Habitualmente no hay que crear ningún contexto gráfico ya que esto es parte del *framework* de AWT
  - Se obtiene mediante el método getGraphics()
- Mediante el método *paint(Graphics contexto)* se determina que es lo que se debe mostrar en dicho contexto

La clase *Graphics*

- Proporciona métodos para dibujar, rellenar, pintar imágenes, copiar áreas y pegar gráficos en pantalla
  - drawLine
  - drawRect y fillRect
  - drawPolygon
  - drawPolyline
  - drawOval y fillOval
  - drawArc y fillArc
- y para escribir texto
  - drawString
  - setFont



Ejemplo gráfico con Canvas (AWT)

```
// canvas que se añade a un frame
public class EjemploCanvas extends Canvas {
        String cad = "Escrito en canvas";
        // este metodo se ejecuta automaticamente cuando Java necesita mostrar la ventana
        public void paint(Graphics g) {
                // obtener el color original
                Color colorOriginal = g.getColor();
                // escribir texto grafico en la ventana y recuadrarlo
                g.drawString(cad, 40, 20);
                g.drawRect(35, 8, (cad.length()*7), 14);
                // dibujo de algunas lineas
                for (int i=20; i< 50; i= i+3) {
                        if ((i % 2) == 0)       g.setColor(Color.blue);
                        else            g.setColor(Color.red);
                        g.drawLine(40, (90-i), 120, 25+i);
                }
                // dibujo y relleno de un óvalo
                g.drawOval(40, 95, 120, 20);
                g.fillOval(40, 95, 120, 20);
                g.setColor(colorOriginal);}}
```