Modulo 3 - Desarrollo de Software basado en Tecnologías Orientadas a Componentes.

Java - Swing

Swing I

Las interfaces gráficas de usuario (GUI) ofrecen al usuario ventanas, cuadros de diálogo, barras de herramientas, botones, listas desplegables y muchos otros elementos con los que ya estamos muy acostumbrados a tratar.

Las aplicaciones son conducidas por eventos y se desarrollan haciendo uso de las clases que para ello nos ofrece la API de Java.

La interfaz de usuario es la parte del programa que permite al usuario interaccionar con él.

La API de Java proporciona una biblioteca de clases para el desarrollo de Interfaces gráficas de usuario (en realidad son dos).

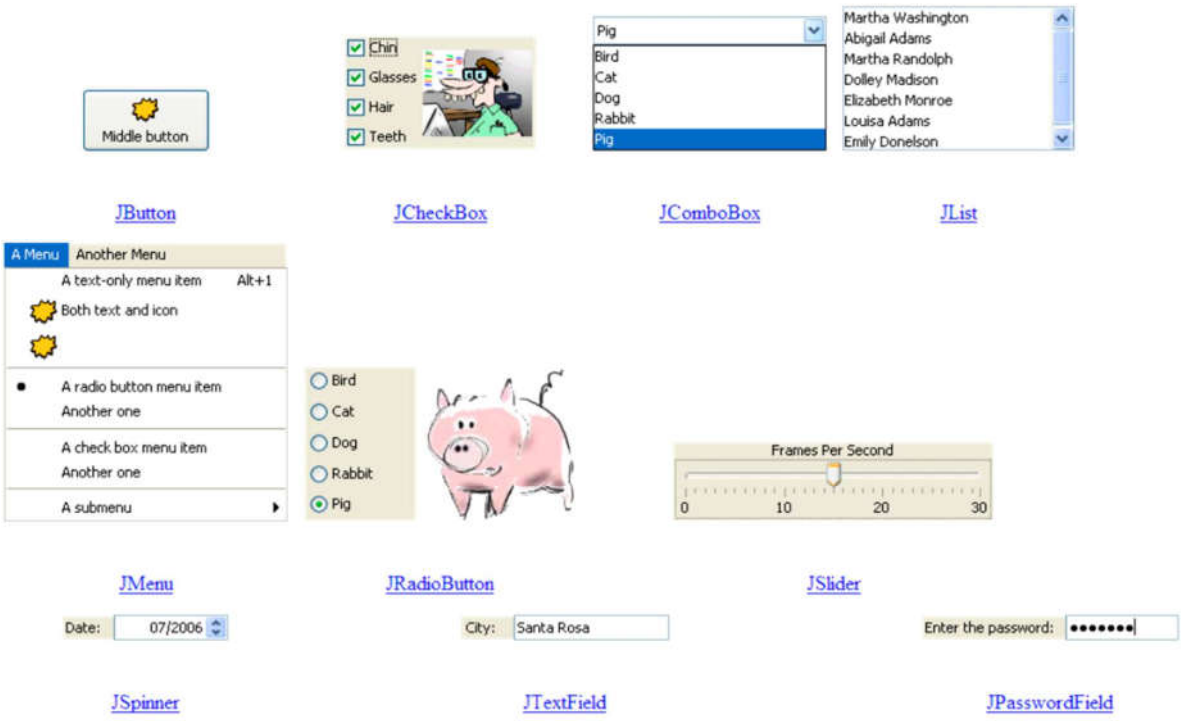
Swing II

La biblioteca proporciona un conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en las que se ejecuten.

La estructura básica de la biblioteca gira en torno a componentes y contenedores. Los contenedores contienen componentes y son componentes a su vez, de forma que los eventos pueden tratarse tanto en contenedores como en componentes.

La API está constituida por clases, interfaces y derivaciones: AWT y Swing,

Componentes Swing I



Jerarquía de clases para las GUI I

Component: superclase de todas las clases de interfaz gráfica.

Container: para agrupar componentes.

JComponent: superclase de todos los componentes de Swing que se dibujan directamente en los lienzos (canvas).

Sus subclases son los elementos básicos de la GUI.

JFrame: ventana que no está contenida en otras ventanas.

JDialog: cuadro de diálogo.

JApplet: subclase de Applet para crear applets tipo Swing.

JPanel: contenedor invisible que mantiene componentes de interfaz y que se puede anidar, colocándose en otros paneles o en ventanas. También sirve de lienzo.

Graphics: clase abstracta que proporciona contextos gráficos donde dibujar cadenas de texto, líneas y otras formas sencillas.

Jerarquía de clases para las GUI II

Color: color de los componentes gráficos.

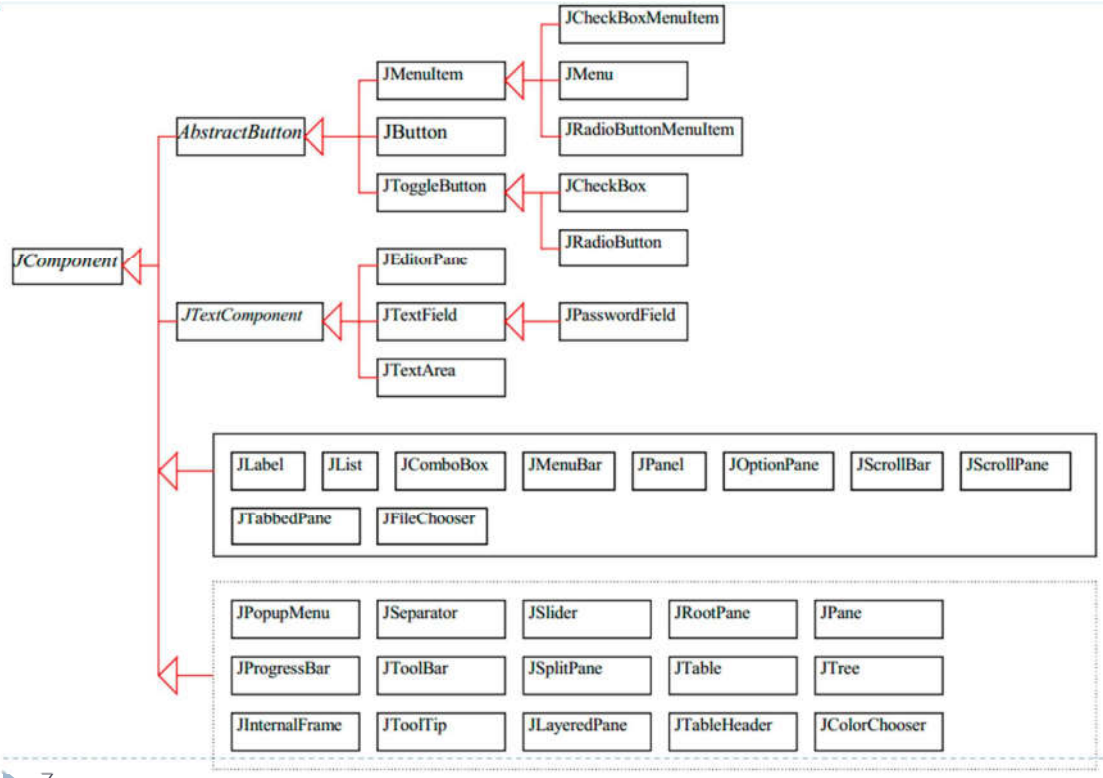
Font: aspecto de los caracteres.

FontMetrics: clase abstracta para propiedades de las fuentes.

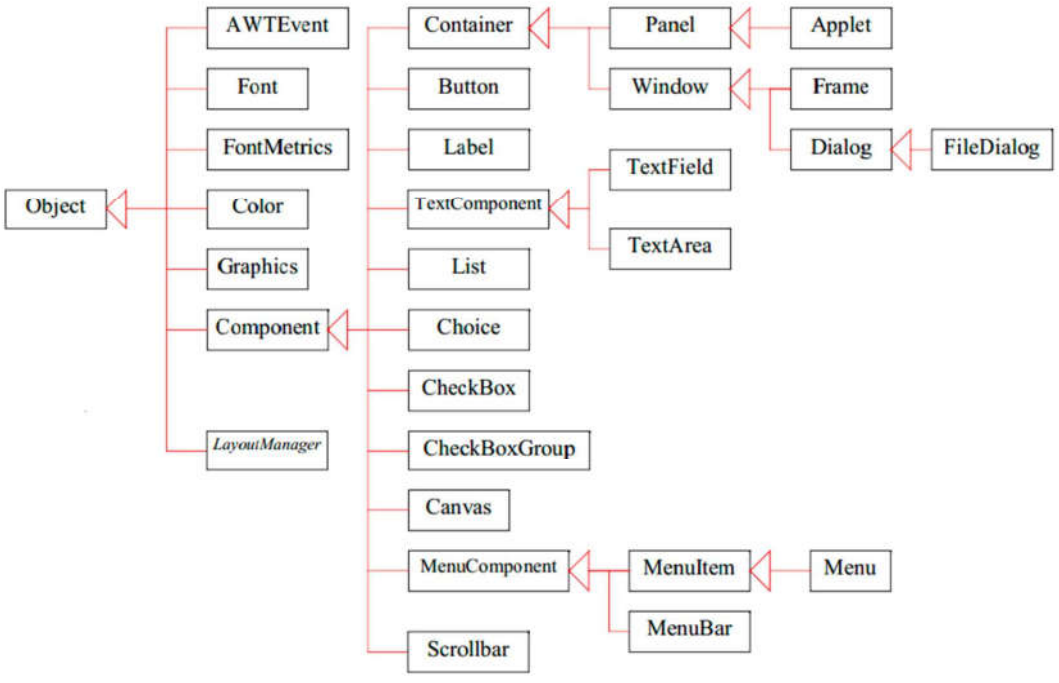
Categorías de clases:

- ✓ Contenedores:
  - ✓ JFrame, JApplet, JWindow, JDialog
- ✓ Componentes intermedios:
  - ✓ JPanel, JScrollPane
- ✓ Componentes:
  - ✓ JLabel, JBbutton, JTextField, JTextArea, ...
- ✓ Clases de soporte:
  - ✓ Graphics, Color, Font, ...

Jerarquía de clases para las GUI: Jcomponent I



Jerarquía de clases para las GUI: Jcomponent II



Jerarquía de componentes I

Graphics (java.awt)
Component (funcionalidad básica de componentes gráficos)
    Button, Canvas, CheckBox, Choice, Label, List, ScrollBar
    TextComponent
        TextField, TextArea
    Container (permite agrupar, añadir y eliminar componentes)
    (también definir diseño o disposición (layout managers))
        ScrollPane
        Panel
            Applet (java.applet)
                JApplet (javax.swing)
        Window
            Frame
                JFrame (javax.swing)
            Dialog
                FileDialog
                JDialog (javax.swing)
            JWindow (javax.swing)
        JComponent (javax.swing)

Contenedores I

Contenedores de alto nivel:

JFrame

Habitualmente la clase JFrame se emplea para crear la ventana principal de una aplicación en Swing.

JDialog

Ventanas de interacción con el usuario.

Contenedores intermedios:

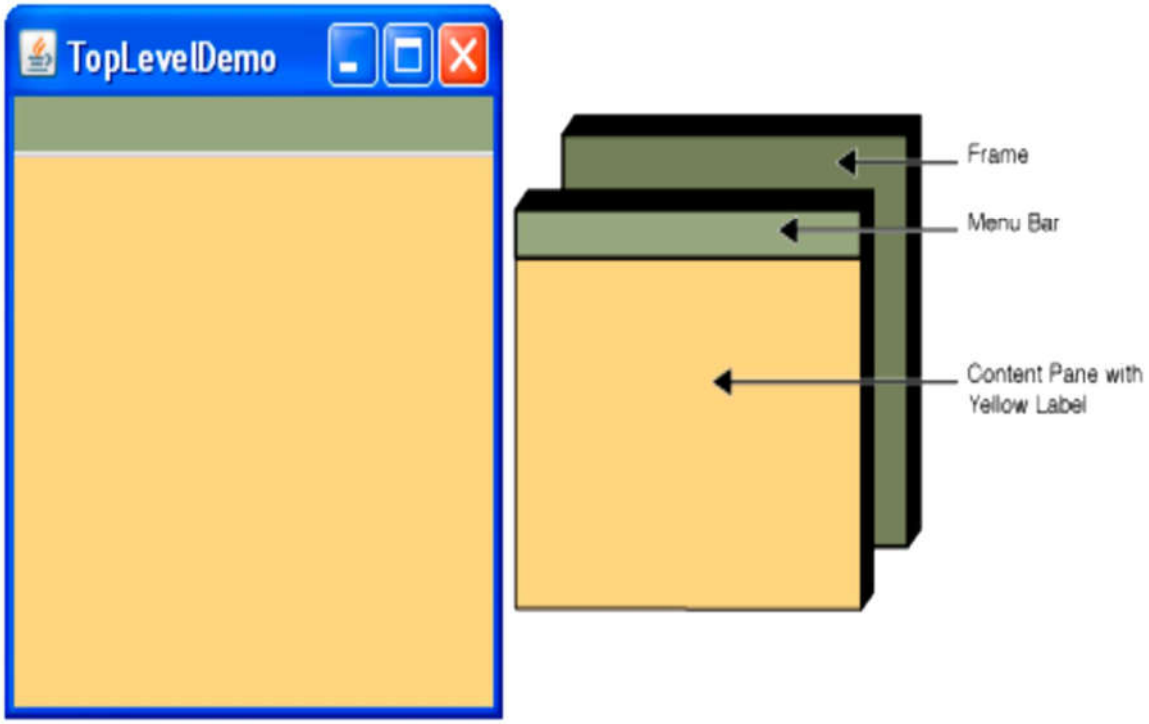
JPanel

Agrupa a otros componentes.

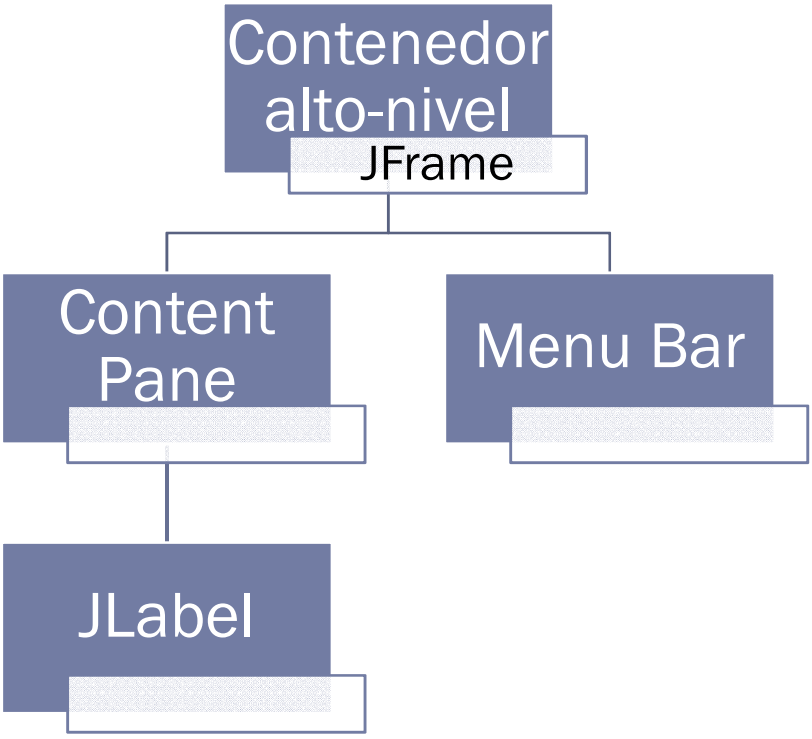
JScrollPanel

Incluye barras de desplazamiento.

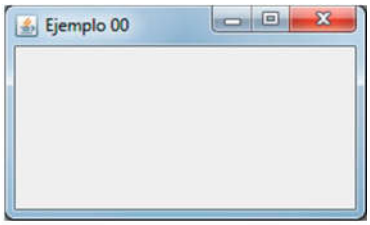
Contenedores II



Jerarquía I



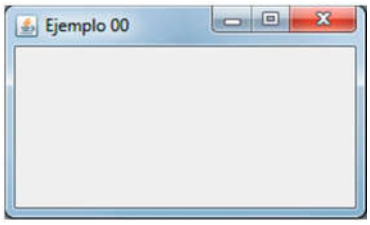
Jerarquía II




```
import javax.swing.*;

public class Gui00 extends JFrame {
// Constantes y componentes (objetos)
    public Gui00() {
        super("Ejemplo 00");
// Configurar Componentes ;
// Configurar Manejadores Eventos ;
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    } // Terminar la aplicación al cerrar la ventana.
    public static void main(String args[]) {
        Gui00 aplicacion = new Gui00();
    }
}
```


Jerarquía III




```
import javax.swing.*;

public class Gui00 {
// Constantes y componentes (objetos)
    public Gui00() {
        Jframe frame = new Jframe("Ejemplo 00");
    // Configurar componentes
    // y añadirlos al panel del frame


    . . .
    frame.pack ();

    frame.setVisible (    true);
    frame.setDefaultCloseOperation (Jframe.EXIT_ON_CLOSE);
    }
    public static void main(String args[]){
        Gui00 aplicacion = new Gui00();
    }
}
```

Añadir elementos

Modo 1:

1. Obtenemos el panel de contenido del frame:

    Container panel = this.getContentPane();

2. Añadimos componentes a dicho panel:

    panel.add(unComponente);

Modo 2:

A partir de 1.5 también se puede hacer directamente sobre el Jframe

    add(unComponente);

1. Con herencia de JFrame utilizando Container

```
import javax.swing.*;
import java.awt.*;

public class Gui01 extends JFrame {

    private Container panel;
    private JButton miboton;

    public Gui01() {
        super("Ejemplo 01 con botón");
// Configurar componentes :
        miboton = new JButton("Aceptar");
        panel = getContentPane();
        panel.add(miboton);

        setSize(200, 100);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui01 aplicacion = new Gui01();
    }
}
```

2. Con herencia de JFrame sin Container

```
import javax.swing.*;
import java.awt.*;

public class Gui01 extends JFrame {

    private JButton miboton;

    public Gui01() {
        super("Ejemplo 01 con botón");
        . . .
        miboton = new JButton("Aceptar");
        add(miboton);
        . . .
        setSize(200, 100);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui01 aplicacion = new Gui01();
    }
}
```

3. Ejemplo sin herencia con Container

```
public class Gui01 {

    private JButton miboton;
    private Container panel;

    public Gui01() {
        Jframe frame = new JFrame(
        "Ejemplo 01");
        panel = frame.getContentPane();
        . . .
        miboton = new JButton("Aceptar");
        panel.add(miboton);
        //se añade al contentPane del frame
        . . .
        frame.pack();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(Jframe.EXIT
    }

    public static void main(String args[]) {
        Gui01 aplicacion = new Gui01();
    }
}
```

4. Ejemplo sin herencia sin Container

```
public class Gui01 {

    private JButton miboton;

    public Gui01() {
        Jframe frame = new JFrame(
        "Ejemplo 01");

    . . .
    miboton  = new JButton("Aceptar");

    frame.add (miboton);  //se añade al frame
    . . .
    frame.pack ();

    frame.setVisible (true);
    frame.setDefaultCloseOperation (Jframe.EXIT_ON_CLOSE);
    }
    public static void main(String args[]) {
        Gui01 aplicacion = new Gui01();
    }
}
```

Administradores de disposición I

Los componentes se agregan al contenedor con el método add().

```
JButton unBoton = new JButton("Texto del botón");
panel.add(unBoton);
```

El efecto de add() depende del esquema de colocación o disposición (layout) del contenedor que se use.

Existen diversos esquemas de disposición: **`FlowLayout, BorderLayout, GridLayout, ...`**

Los objetos contenedores se apoyan en objetos **`LayoutManager`** (administradores de disposición).

Clases más usadas que implementa la interfaz **`LayoutManager:`**

`FlowLayout:` un componente tras otro de izquierda a derecha.
`BorderLayout:` 5 regiones en el contenedor (North, South, ...).
`GridLayout:` contenedor en filas y columnas.

Administradores de disposición II

Para organizar el contenedor se utiliza el método setLayout():

```
public void setLayout(LayoutManager mgr)
```

Crea una disposición para el contenedor actual

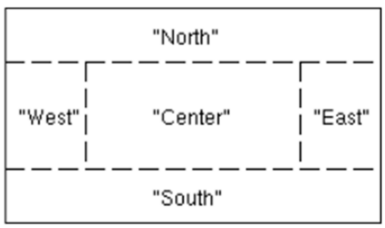
Tipo de disposición (BorderLayout, ...)

```
setLayout(new BorderLayout());
setLayout(new FlowLayout());
setLayout(new GridLayout(3,4));
```

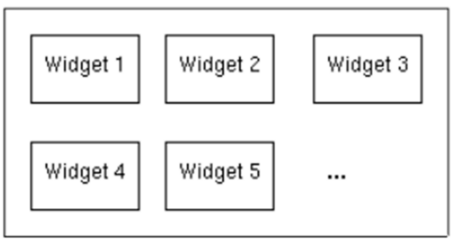
El layout manager elige la mejor posición y tamaño de cada componente de acuerdo al espacio disponible.

Administradores de disposición III

BorderLayout organiza el contenedor en 5 zonas:



FlowLayout coloca los componentes de izquierda a derecha y de arriba hacia abajo:



Para distribuciones más complejas podemos insertar paneles (JPanel) en los contenedores y obtener el tamaño de un componente con el método getSize().

FlowLayout

```
public class Gui02 extends JFrame {

    public Gui02() {
        super("Ejemplo de Layout");
        // Configurar componentes ;
        // Configurar layout ;
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
        for (int i = 1; i <= 10; i++) {
            add(new JButton("Componente " + i));
        }
        setSize(200, 200);//pack();
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui02 aplicacion = new Gui02();
    }
}
```

BorderLayout

```
public class Gui03 extends JFrame {

    public Gui03() {
        super("Ejemplo de Layout");
        // BorderLayout
        setLayout(new BorderLayout(5, 10));
        add(new JButton("1"), BorderLayout.EAST);
        add(new JButton("2"), BorderLayout.SOUTH);
        add(new JButton("3"), BorderLayout.WEST);
        add(new JButton("4"), BorderLayout.NORTH);
        add(new JButton("5"), BorderLayout.CENTER);
        setSize(200, 200); //pack();
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    public static void main(String args[]) {
        Gui03 aplicacion = new Gui03();
    }
}
```

GridLayout

```
setLayout(new GridLayout(filas, columnas))
```

Crea una zona de filas x columnas componentes y éstos se van acomodando de izquierda a derecha y de arriba a abajo.

GridLayout tiene otro constructor que permite establecer la separación (en pixels) ente los componentes, que es cero con el primer constructor.

Así, por ejemplo:

```
new GridLayout(3, 4, 2, 2)
```

crea una organización de 3 filas y 4 columnas donde los componentes quedan a dos pixels de separación.

Ejemplo:

```
setLayout(new GridLayout(3, 4, 2, 2);
for(int i = 0; i < 3 * 4; i++) {
   add(new JButton(Integer.toString(i + 1)));
}
```


GridLayout

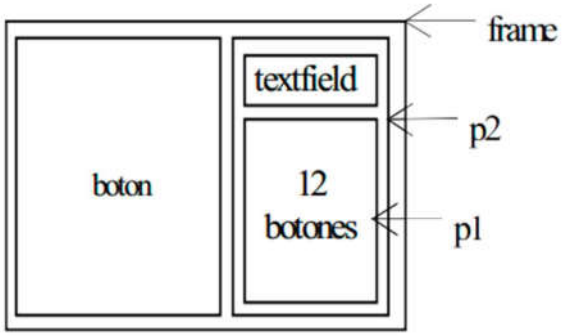
```
public class Gui03b extends JFrame {

    public Gui03b() {
        super("Ejemplo de Layout");
        setLayout(new GridLayout(4, 3, 5, 5));
        for (int i = 1; i <= 10; i++) {
            add(new JButton(Integer.toString(i)));
        }
        setSize(200, 200);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui03b aplicacion = new Gui03b();
    }
}
```

Paneles como contenedores I

Los paneles actúan como pequeños contenedores para agrupar componentes. Colocamos los componentes en paneles y los paneles en el frame o incluso en otros paneles.



Paneles como contenedores II

```
public class Gui04 extends JFrame {

    public Gui04() {
        setTitle("Horno microondas");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        // Create panel p1 for the buttons and set GridLayout
        JPanel p1 = new JPanel();
        p1.setLayout(new GridLayout(4, 3));
        // Add buttons to the panel
        for (int i = 1; i <= 9; i++) {
            p1.add(new JButton("" + i));
        }
        p1.add(new JButton("" + 0));
        p1.add(new JButton("Start"));
        p1.add(new JButton("Stop"));
```

Paneles como contenedores III

```
        // Create panel p2 to hold a text field and p1
        JPanel p2 = new JPanel();
        p2.setLayout(new BorderLayout());
        p2.add(new JTextField("Aquí el tiempo"),
                BorderLayout.NORTH);
        p2.add(p1, BorderLayout.CENTER);
        // Add p2 and a button to the frame
        add(p2, BorderLayout.EAST);
        add(new Button("Aquí la comida"),
                BorderLayout.CENTER);
        setSize(400, 250);
        setVisible(true);
    }

    public static void main(String[] args) {
        Gui04 frame = new Gui04();
    }
}
```


Dibujo de gráficos en paneles

JPanel se puede usar para dibujar.

Para dibujar en un panel se crea una clase derivada de Jpanel y se redefine el método paintComponent() que le indica al panel como dibujar.

La clase ***Graphics*** es una clase abstracta para todos los contextos gráficos.

Una vez obtenido el contexto gráfico podemos llamar desde este objeto a las funciones gráficas definidas en la clase Graphics.

Graphics contiene información acerca de la zona que necesita ser redibujada: el objeto donde dibujar, un origen de traslación, el color actual, la fuente actual, etcétera.