

GUI

Interfase gráfica del usuario

Interfase del usuario

- **La interfase de usuario** es la parte del programa que permite a éste interactuar con el usuario. Las interfaces de usuario pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas.
- **La interfase de usuario** es el **aspecto más importante de cualquier aplicación**. Una aplicación sin un interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa. Java proporciona los elementos básicos para construir *decentes* interfaces de usuario a través del AWT.

El java.awt tiene todos los elementos para crea una interfase gráfica

Dentro de una interfase gráfica tenemos los siguiente:

- Componentes
- Contenedores
- Layouts (Administradores de diseño)
- Eventos

- **Componentes.** Son clases o interfaces que permiten crear los objetos gráficos que componen una GUI tales como; botones, listas desplegables, cuadros de texto, casillas de verificación, botones de opción, campos de texto, etiquetas, menús, etc.
- **Contenedores.** Son clases o interfaces que permiten crear objetos gráficos para contener a los componentes, tales como; paneles, cuadros de diálogo, marcos, ventanas, etc.

- **Layouts.** Son clases o interfaces que permiten crear objetos de que administren el diseño , la distribución y colocación de los objetos componentes dentro de los objetos contenedores. Por ejemplo el **FlowLayout** distribuye los componentes de izquierda a derecha y de arriba a abajo, el **BorderLayout** los distribuye en cinco áreas geográficas; norte, sur, este, oeste y centro, etc..
- **Eventos.** Son las clases o interfaces que permiten crear objetos que capturen y manejen los eventos. Un evento es una acción sobre algún componente, por ejemplo, clic a un botón, pulsar la tecla de enter en un botón, mover un elemento con las teclas de navegación, eventos especiales como los programados por tiempo, etc. Sin los eventos las GUI serían interfaces gráficas sin vida, y por lo tanto no serían muy útiles que digamos.

Componentes

Tabla. Las AWT y sus componentes

- **Tipo de Componente**
- **Button** Es un botón usado para recibir el clic del ratón
- **Canvas** Un lienzo o panel usado para dibujar
- **Checkbox** Cuadro de verificación. Es un componente que le permite seleccionar un elemento
- **CheckboxMenuItem** Es un cuadro de verificación dentro de un menú
- **Choice** Es una lista desplegable de elementos estáticos.

Tipos de componentes

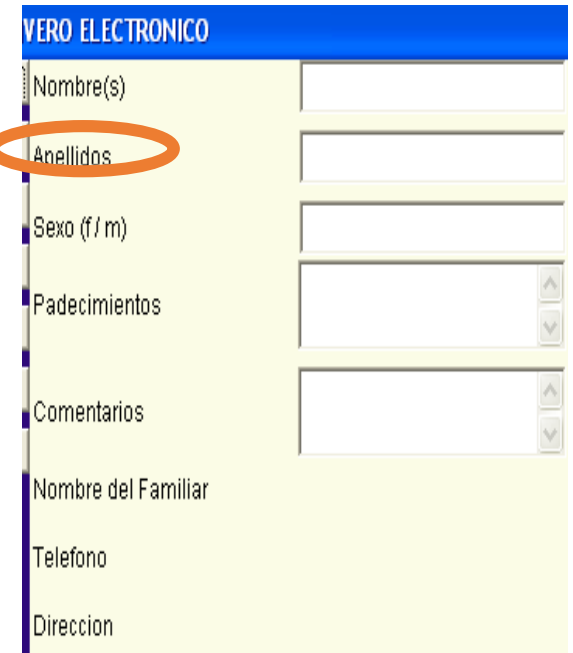
- **Component** Es el padre de todos los componentes AWT, excepto de los componentes de tipo
- **menúContainer** Es el padre de todos los contenedores
- **Dialog** Es un cuadro de diálogo o una ventana de alto nivel con título y bordes.
- **Frame** Es un marco o ventana y es la clase base de todas las ventanas GUI con controles para ventana.
- **LabelEtiqueta.** Es una cadena de texto como componente.
- **List** Un componente que contiene un conjunto dinámico de elementos.
- **Menu** Es un elemento dentro de la barra de menú, el cual contiene un conjunto de elementos de tipo menú.

Tipos de componentes

- **MenuItem** Un elemento dentro de un menú.
- **Panel** Una clase contenedora básica usado frecuentemente para crear diseños (layouts) complejos.
- **Scrollbar** Un componente que permite al usuario hacer una selección dentro de un rango de valores
- **ScrollPane** Una clase contenedora que implementa un deslizador horizontal y vertical para un único componente hijo
- **TextArea** Un componente que permite al usuario introducir texto en un bloque o rectángulo.

Creando etiquetas

- **Label** sirve para crear etiquetas o letreros descriptivos



VERO ELECTRONICO

Nombre(s)	<input type="text"/>
Apellidos	<input type="text"/>
Sexo (f / m)	<input type="text"/>
Padecimientos	<input type="text"/>
Comentarios	<input type="text"/>
Nombre del Familiar	<input type="text"/>
Telefono	<input type="text"/>
Direccion	<input type="text"/>

Formato general:

Label objetoTipoEtiqueta; // definición

objetoTipoEtiqueta = new Label ("etiqueta"); // creación

Creando etiquetas

Constructores

```
public label (String text)
```

```
public label (String text, int alignmet)
```

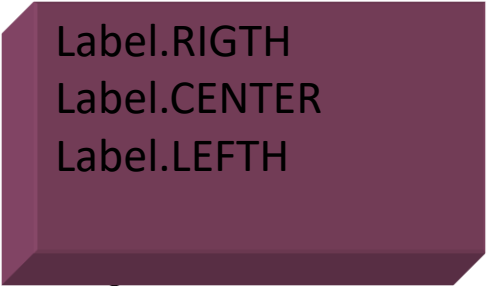
- Ejemplos:

```
Label nomLabel; // define nomLabel como objeto de la clase Label
```

```
nomLabel = new Label("Nombre:");
```

```
Label claveLabel;
```

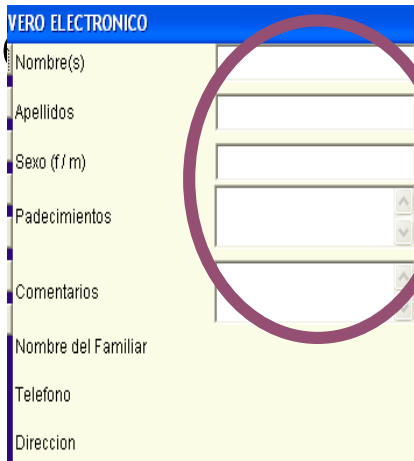
```
claveLaber=new Label("Clave", Label.CENTER);
```



Label.RIGTH
Label.CENTER
Label.LEFTH

Creando un campo de Texto

Campos de Texto: Permiten leer y
muestra una sola línea de texto.



VERO ELECTRONICO

Nombre(s)	<input type="text"/>
Apellidos	<input type="text"/>
Sexo (f / m)	<input type="text"/>
Padecimientos	<input type="text"/>
Comentarios	<input type="text"/>
Nombre del Familiar	<input type="text"/>
Telefono	<input type="text"/>
Direccion	<input type="text"/>

Formato general:

TextField objeto;

objeto = new TextField(#caracteres);

Creando un campo de Texto

Constructores

```
public TextField(String Texto)
public TextField(String text)
public TextField(int length)
public TextField(String text, int length)
```

Ejemplos:

- `TextField nomText; //` define **nomText** como objeto de la clase `TextField`
- `nomText = new TextField(15);`

Metodos utiles:

- `setEditable(boolean)` define si el texto es editable o no
- `select(int, int)` selecciona el texto entre las dos posiciones (origen = 0)
- etc...

Creando un área de texto

Área de Texto que permite leer y escribir en pantalla varias líneas de texto.

-Formato general:

TextArea objeto;

objeto = new TextArea(#caracteres);



The screenshot shows a window with a light gray background. It contains two text areas. The first text area is labeled "Address:" and contains the text "JPE Associates", "341 Lafayette Street", "Suite 25", and "NY NY 10012". The second text area is labeled "Instructions:" and contains the text "This is a mailbox rental store. Please don't actually look for me there". Both text areas have a vertical scrollbar on the right side and a horizontal scrollbar at the bottom.

Área de texto

Constructores

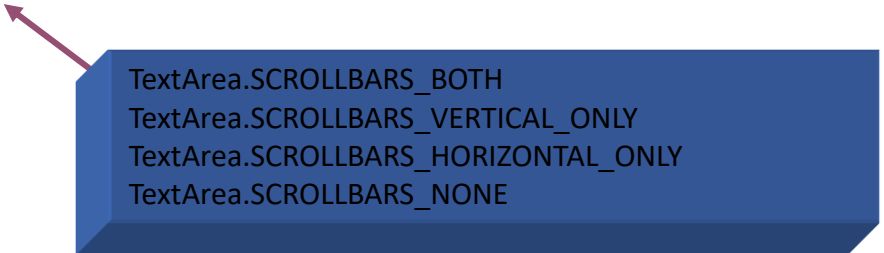
```
public TextArea()
```

```
public TextArea(String text)
```

```
public TextArea(int rows, int columns)
```

```
public TextArea(String text, int rows, int columns)
```

```
public TextArea(String text, int rows, int columns, int  
scrollbars)
```



```
TextArea.SCROLLBARS_BOTH  
TextArea.SCROLLBARS_VERTICAL_ONLY  
TextArea.SCROLLBARS_HORIZONTAL_ONLY  
TextArea.SCROLLBARS_NONE
```

Área de Texto

Ejemplo:

```
TextArea address = new TextArea("Type your address here", 5, 80);
```

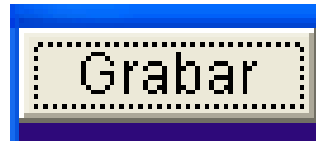
```
TextArea instructions = new TextArea("", 15, 70,  
    TextArea.SCROLLBARS_VERTICAL_ONLY);
```

Metodos utiles:

`insertText(String, int)` // inserta texto en la posicion indicada
`replaceText(String, int,int)` // reemplaza el texto entre las posiciones dadas

Crear Botones

- Button.- Clase definida para crear botones, para interactuar con el usuario.



Formato General:

```
Button objetoBoton;
```

```
objetoBoton = new Button("etiquetaDelBoton");
```

Creando Botones

- Ejemplos

Button b; // define **b** como objeto de la clase Button

b = new Button("CALCULA");

Button C;

C = new Button("Mi primer botón");

Listas

```
Formato General:  
List objetoLista;  
objetoLista = new List();
```

Ejemplo:


```
List l; // define l como objeto de la clase List  
l = new List();
```

Constructores

```
public List()
```

```
public List(int numLines)
```

```
public List(int numLines, boolean  
allowMultipleSelections)
```



```
true. Selección múltiple permitida.  
false. Selección múltiple no permitida.
```

Añadiendo el elemento

add

Añade el elemento gráfico a la ventana `add(objeto);`

Ejemplo:

```
add(t1);
```

Contenedores

Contenedores

- Los Componentes no se encuentran aislados, sino agrupados dentro de *Contenedores*. Los Contenedores **contienen y organizan la situación de los Componentes**; además, los contenedores son en sí mismos componentes y como tales pueden ser situados dentro de otros contenedores

Panel

Un panel es un contenedor de elementos de cualquier tipo, incluyendo otros paneles y puede tener un manejador de layout definido.

Los paneles se usan para agrupar elementos dentro de otros contenedores.

Panel

```
Panel objetoPanel;  
objetoPanel = new Panel(objeto Layout);
```

Ejemplo:

Panel p; // define **p** como objeto de la clase Panel

p = new Panel(new GridLayout(3, 3, 0, 0));

Creando un Panel

El nuevo Panel tendrá un Layout de 3 por 3
elementos

Constructores

```
public Panel()
```

```
public Panel(manejador de Layout layout)
```

Layouts

Manejadores de Layout

El acomodo de los componentes gráficos en la ventana, lo hace el manejador de layout.

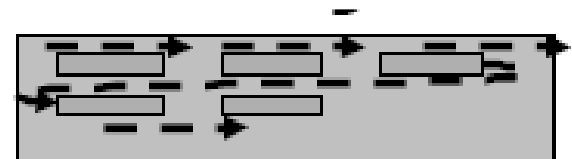
Esto permite crear interfaces rápidamente sin necesidad de calcular posiciones exactas ni tamaños.

- Existen 3 tipos de manejadores
- 1. FlowLayout
- 2. GridLayout
- 3. BorderLayout

Layouts

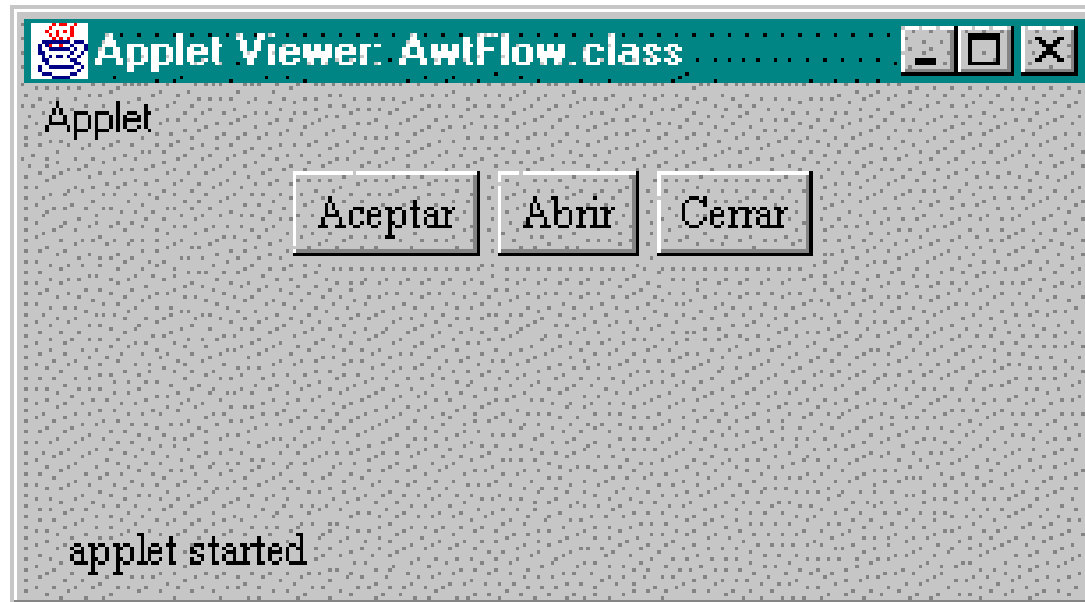
Especifican la apariencia que tendrán los componentes a la hora de colocarlos sobre un Contenedor:

- **FlowLayout** Es el más simple y el que se utiliza por defecto en todos los Paneles si no se fuerza el uso de alguno de los otros. Los Componentes añadidos a un Panel con FlowLayout se encadenan en forma de lista. La cadena es horizontal, de izquierda a derecha, y se puede seleccionar el espaciado entre cada Componente.



Layouts

Grupo de botones con la composición por defecto que proporciona
FlowLayout:



FlowLayout

Acomoda los elementos de izquierda a derecha en el orden en el que fueron agregados (add). Cuando se llega a la orilla de la derecha, continúa en el siguiente renglón.

Constructores

```
public FlowLayout()
```

```
public FlowLayout(int alineación)
```

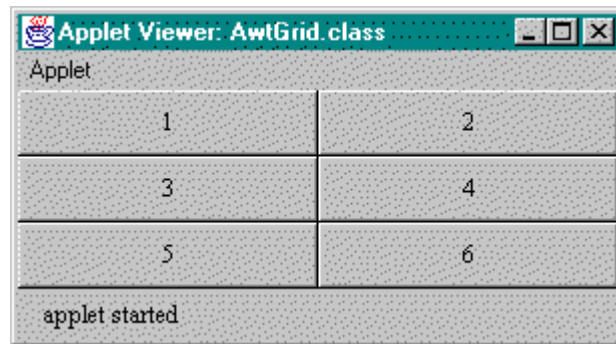
```
public FlowLayout(int alineación int hgap int vgap)
```

- Ejemplo

```
setLayout(new FlowLayout(FlowLayout.LEFT), 10, 10); // alineacion  
izquierda, con distancias de 10 pixeles en horizontal y vertical
```


Layouts

- **GridLayout** La composición GridLayout proporciona gran flexibilidad para situar Componentes. El layout se crea con un número de filas y columnas y los Componentes van dentro de las celdas de la tabla así definida.



GridLayout

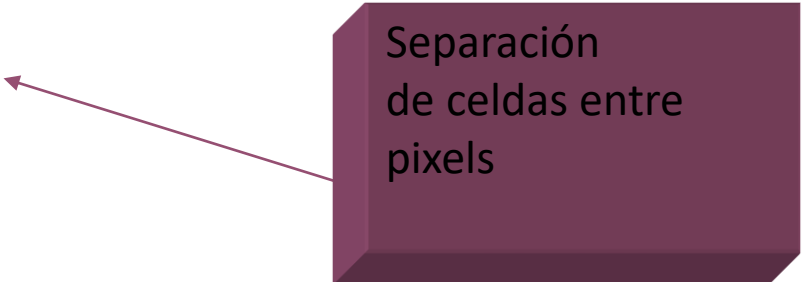
- Este manejador de layout divide la ventana en una cuadrícula (grid) de manera que los elementos se colocan
- en renglones y columnas. Todos los elementos de un GridLayout miden el mismo ancho y largo.
- Los elementos se agregan empezando por la parte superior izquierda hacia la derecha; cuando se llena el renglón, continúa de la misma forma en el siguiente renglón.

Constructores

```
public GridLayout()
```

```
public GridLayout(int renglon, int columnas)
```

```
public GridLayout(int renglones, int columnas, int hgap, int vgap)
```



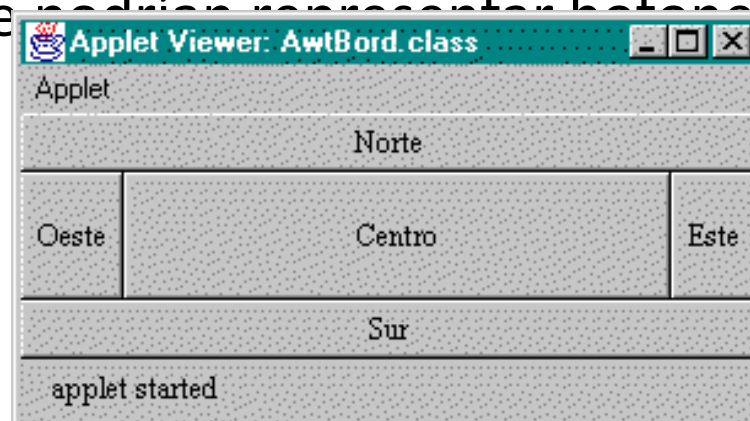
Separación
de celdas entre
pixels

- Ejemplo

`setLayout(new GridLayout(3,3));` // Layout de 3 filas y 3 columnas

Layouts

- **BorderLayout** La composición BorderLayout (de borde) proporciona un esquema más complejo de colocación de los componentes en un panel.
- La composición utiliza cinco zonas para colocar los componentes sobre ellas: **Norte, Sur, Este, Oeste y Centro**. Es el layout o composición que se utilizan por defecto Frame y Dialog.
- Con BorderLayout se podrían representar botones de dirección:



- Se usa add(elemento, posición)
donde posición puede ser BorderLayout.NORTH
BorderLayout.SOUTH
BorderLayout.EAST
BorderLayout.WEST
BorderLayout.CENTER

Constructores

```
public BorderLayout()
```

```
public BorderLayout(int hgap, int vgap)
```

Obteniendo datos numéricos

Usar método `getText()` de la clase `TextField`

Usar `Integer.parseInt()` para entero

Usar `Double.parseDouble()` para números reales

Para extraer un valor numérico de un texto, es necesario utilizar el método **getText** de la clase **TextField**, así como el método **parseInt** de la clase **Integer**.

El método **getText** toma de un objeto de tipo **TextField** el valor y lo convierte a objeto tipo **String**. El método **parseInt** toma el valor **String** y lo convierte a un valor entero primitivo

Para extraer un valor numérico de un texto, es necesario utilizar el método **getText** de la clase **TextField**, así como el método **parseInt** de la clase **Integer**

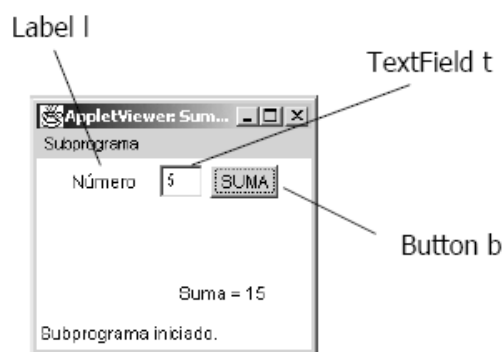


Diagram illustrating the code snippet: `numero = Integer.parseInt (t.getText());`. The code is shown in a gray box. Annotations indicate that `t` is an "objeto TextField" and `t.getText()` is an "objeto String". A horizontal line below the code, with a vertical line pointing down to the text "valor primitivo entero", indicates that the result of the `parseInt` method is a primitive integer value.

Desplegando información

Para dejar un valor en un campo de texto se utiliza el método `setText`.

El `setText` utiliza un objeto `String` como parámetro para definir el valor del texto.

Ejemplos:

```
t.setText("Error en los datos");
```

```
t.setText("Valor = " + x );
```

Eventos

Con respecto a tu computadora un evento podría ser:

Que el usuario movió el mouse,

Que el usuario hizo click en un botón,

que recorrió el scrollbar, o que el usuario presionó alguna tecla, por ejemplo: \neg , , [®] , -

Para manejar eventos a través de botones es necesario:

Implementar el escuchador. Paquete que tendrá la responsabilidad de escuchar a tus botones.	<pre>public class Colores extends Applet implements ActionListener</pre>
Agregar el botón y dar a cada botón la “facultad” de ser escuchados cuando el usuario haga click en ellos.	<pre>add(boton1); boton1.addActionListener(this);</pre>
Manejar el evento. (Si tienes varios botones se debe preguntar qué botón fue presionado. Más adelante veremos ejemplos que utilicen varios botones.)	<pre>public void actionPerformed(ActionEvent e) (En el momento que el usuario hace click en un botón, el control del programa se transfiere a este método).</pre>