2. Interfaces gráficas de usuario

Para iniciar tenga en cuenta la interfaz de programación de aplicaciones, más conocida como Application Programming Interface (API), enfocada en la programación de objetos. API es un conjunto de funciones y procedimientos organizados en una biblioteca para ser usados por otro software como una capa de abstracción; para el caso del API Java, su biblioteca cuenta con dos clases, AWT y Swing.

Tema 1. Elementos de una interfaz

Una interfaz gráfica debe contener elementos en los que se puedan definir etiquetas, campos de texto, botones, barras deslizadoras, entre otros.

A continuación se muestran algunos elementos de una interfaz gráfica, así como las diferentes maneras de utilizarlos:

Etiqueta (Label)

La etiqueta ayuda a definir los campos de texto que tecleará el usuario; la clase definida para esto es Label, asimismo existen diferentes maneras de construir una:

- Label ()
- Label (String str)
- Label (String str, int alignment)

Constructor summary	
Label ()	Constructs an empty label.
Label (String text)	Constructs a new label whit the specified string of text, left justified.
Label (String text, int alignment)	Constructs a new label that presents the specified string of text whit the specified alignment.

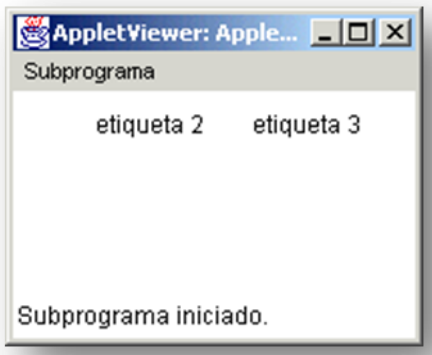
El primer constructor crea una etiqueta en blanco, el segundo crea una con el string mandado como parámetro y el tercero crea una etiqueta con el string mandado, definiendo por medio del how la manera en la que se alineará esta etiqueta, siendo how: Label.LEFT, Label.RIGHT o Label. CENTER.

Un ejemplo del uso de etiquetas puede ser:

```
import java.awt.*;
import java.applet.*;
// <applet width="200" height="100" code="AppletInterfaz1"></applet>
public class AppletInterfaz1 extends Applet {
 Label l1, l2, l3;
 public AppletInterfaz1() {
 l1 = new Label();
 l2 = new Label("etiqueta 2");
 l3 = new Label("etiqueta 3", Label.CENTER);
 add(l1);
 add(l2);
 add(l3);
 }
}
```

Mostrando el applet:

Applet etiquetas



Fuente: SENA

Es difícil entender que existe una etiqueta en blanco allí, y que la tercera etiqueta está alineada al centro. Más adelante se muestra de nuevo este applet, pero utilizando clases que pueden servir para mejorar la interfaz haciendo uso de administradores de distribución llamados layout managers.

Puede quedar la duda de si ya se creó l1 como un objeto de la clase label, pero sin etiqueta, sin embargo esto es posible cambiarse a través del método setText(), de tal manera que se utilizaría l1.setText("etiqueta 1").

Campo de texto (Text field)

Un campo texto sirve para introducir un dato del usuario a la aplicación, esto puede hacerse de diferentes maneras:

TextField(), TextField(int numChars), TextField(String str), TextField(String str, int numChars).

Constructor summary	
TextField ()	Constructs a new text field.
TextField (int columns)	Constructs a new empty text field with the specified number of columns.
TextField (String text)	Constructs a new text field initialized with the specified text.
TextField	Constructs a new text field initialized with the specified text to be displayed, and wide enough to hold the specified number of columns.

El primer constructor crea un campo texto, el segundo constructor sirve para definir el número de caracteres que se quieren ver al introducir algún dato; el tercer constructor ayuda a determinar algún texto inicial que se desea que aparezca al usuario, y el último constructor es para usar ambas opciones.

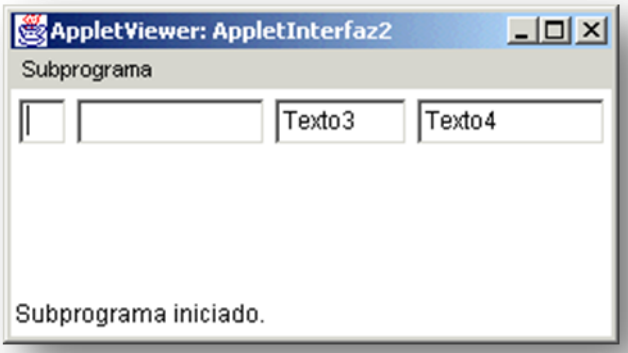
Un ejemplo del uso de estos constructores es:

```
import java.awt.*;
import java.applet.*;
// <applet width="300" height="100" code="AppletInterfaz2"></applet>
public class AppletInterfaz2 extends Applet {
 TextField t1, t2, t3, t4;
 public AppletInterfaz2() {
 t1 = new TextField();
 t2 = new TextField(10);
 t3 = new TextField("Texto3");
 t4 = new TextField("Texto4",10);
 add(t1);
 add(t2);
 add(t3);
 add(t4);
```

```
 }
}
```

Y la manera de visualizar esto sería:

Applet campos de texto



Fuente: SENA

De esta forma se observa como el segundo y cuarto texto son más grandes debido al tamaño que se puso al constructor.

Botón (Button)

Con esta clase se define algún botón para que el usuario por medio de alguna acción pueda escuchar el evento.

La forma de crear botones son: Button (), Button(String str).

Constructor summary	
Button()	Constructs a button with an empty string for its label.
Button (String label)	Constructs a button with the specified label.

Crear un botón sin etiqueta o con etiqueta

La manera de cambiar la etiqueta a un botón o de asignar una si no la tiene sería utilizando el método setLabel().

Una aplicación que muestra el uso de estos dos constructores es:

```
import java.awt.*;
import java.applet.*;
// <applet width="200" height="100" code="AppletInterfaz3"></applet>
public class AppletInterfaz3 extends Applet {
 Button b1, b2;
 public AppletInterfaz3() {
 b1 = new Button();
 b2 = new Button("boton2");
 add(b1);
 add(b2);
 }
}
```

La forma de visualizar esto es:

Applet botones

Fuente: SENA

De esta manera se revisan las dos formas de crear un botón.

Opciones (Choice)

Esta clase ayuda a utilizar menús de opciones donde el usuario fácilmente puede seleccionar una alternativa. La manera de hacer esto es con el constructor:

Choice().

Constructor summary	
Choice()	Create a new choice menu.

Al crear un objeto de este tipo se le añaden las opciones que se desea que aparezcan en el menú, utilizando el método add() o addItem().

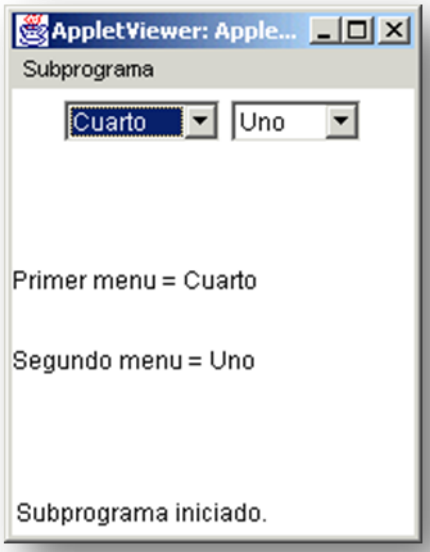
Para saber qué valor se seleccionó se usa el método getSelectedItem() o el getSelectedIndex(); el primero toma el string del elemento escogido y el segundo da el índice. Ejemplo de una aplicación de este elemento gráfico:

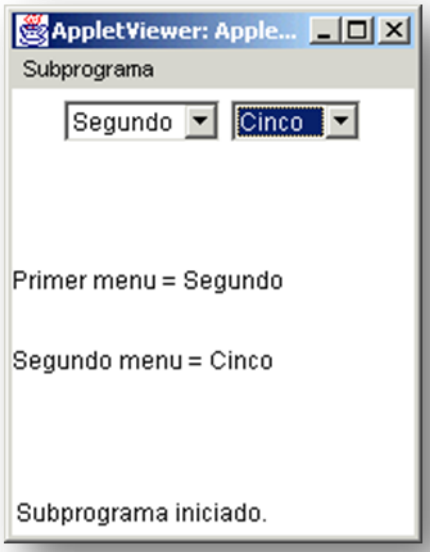
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
// <applet width="200" height="200" code="AppletInterfaz4"></applet>
public class AppletInterfaz4 extends Applet implements ItemListener{
 Choice ch1, ch2;
 public AppletInterfaz4() {
 ch1 = new Choice();
 ch2 = new Choice();
 ch1.add("Primero");
 ch1.add("Segundo");
 ch1.add("Tercero");
 ch1.add("Cuarto");
 ch2.add("Uno");
 ch2.add("Dos");
 ch2.add("Tres");
 ch2.add("Cuatro");
 ch2.add("Cinco");
 ch1.select("Cuarto");
 add(ch1);
 add(ch2);
 ch1.addItemListener(this);
 ch2.addItemListener(this);
 }

 public void itemStateChanged(ItemEvent ie) {
 repaint();
 }
```

```
 public void paint(Graphics g) {
 String msg = "Primer menu = ";
 msg += ch1.getSelectedItem();
 g.drawString(msg, 0,100);
 msg = "Segundo menu = ";
 msg += ch2.getSelectedItem();
 g.drawString(msg, 0,140);
 }
}
```

Algunos ejemplos utilizando este applet serían (inicial y seleccionando otros valores):

Applet menús





Fuente de imágenes: SENA

Además de los métodos mencionados para esta clase están el getItemCount() que da el número de elementos en la lista y hay otros más, los cuales es conveniente revisar en la clase choice.

Lista (List)

Esta clase provee una compacta lista de múltiples opciones con el uso de la barra deslizadora. Los constructores de ésta son:

- List()
- List(int lineas)
- List (int lineas, boolean seleccionmultiple)

Constructor summary	
List ()	Creates a new scrolling list.
List (int rows)	Creates a new scrolling list initialized with the specified number of visible lines.
List (int rows, boolean multipleMode)	Creates a new scrolling list initialized to display the specified number of rows.

El primer constructor crea una lista con el poder de selección de un dato a la vez, el segundo especifica el número de líneas que serán visibles al mismo tiempo, y el tercero con la opción de selección múltiple en true permitirá que se pueda elegir más de un elemento a la vez. El método add() se emplea para añadir elementos a la lista, pero tiene dos opciones, la de agregar un string, o la de incorporar un string en cierta posición.

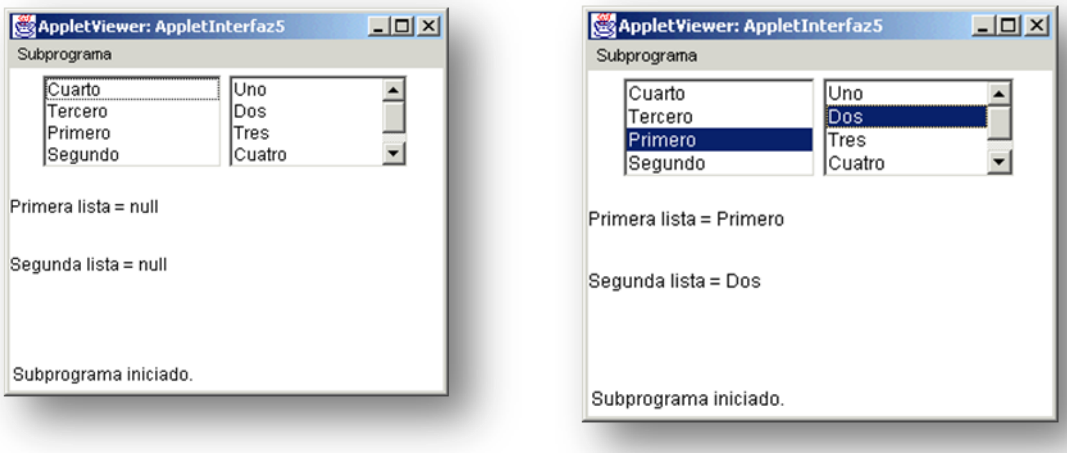
La siguiente es una aplicación que lo utiliza:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
// <applet width="300" height="200" code="AppletInterfaz5"></applet>
public class AppletInterfaz5 extends Applet implements ItemListener{
 List ch1, ch2;
 public AppletInterfaz5() {
 ch1 = new List();
 ch2 = new List();
 ch1.add("Primero");
 ch1.add("Segundo");
 ch1.add("Tercero",0);
 ch1.add("Cuarto",0);
 ch2.add("Uno");
 ch2.add("Dos");
 ch2.add("Tres");
 ch2.add("Cuatro");
 ch2.add("Cinco");
 add(ch1);
```

```
add(ch2);
ch1.addItemListener(this);
ch2.addItemListener(this);
}

public void itemStateChanged(ItemEvent ie) {
repaint();
}

public void paint(Graphics g) {
String msg = "Primera lista = ";
msg += ch1.getSelectedItem();
g.drawString(msg, 0,100);
msg = "Segunda lista = ";
msg += ch2.getSelectedItem();
g.drawString(msg, 0,140);
}
}
```

Lo anterior se mostrará de la siguiente manera (al inicio y con valores seleccionados):

Applet listas



Fuente de imágenes: SENA

Área de texto (Text area)

Este elemento sirve para tomar o desplegar datos que tendrán más de una línea.

Existen varios constructores:

- TextArea()
- TextArea(int lineas, int caracteres)
- TextArea(String str)
- TextArea(String, str, int lineas, int caracteres)
- TextArea(String str, int lineas, int caracteres, int barra)

Constructor summary	
TextArea ()	Constructs a new text area with the empty string as text.
TextArea (int rows, int columns)	Constructs a new text area with the specified number of rows and columns and the empty string as text.
TextArea (String text)	Constructs a new text area with the specified text.
TextArea (String text, int rows, int columns)	Constructs a new text area with the specified text, and with the specified number of rows and columns.
TextArea (String text, int rows, int columns, int scrollbars)	Constructs a new text area with the specified text, and with the rows, columns, and scroll bar visibility as specified.

Líneas es el número de líneas, caracteres es el número de caracteres, str es el string que se desea que aparezca y la barra puede ser una de las siguientes:

- SCROLLBARS_BOTH
- SCROLLBARS_HORIZONTAL_ONLY
- SCROLLBARS_VERTICAL_ONLY
- SCROLLBARS_NONE

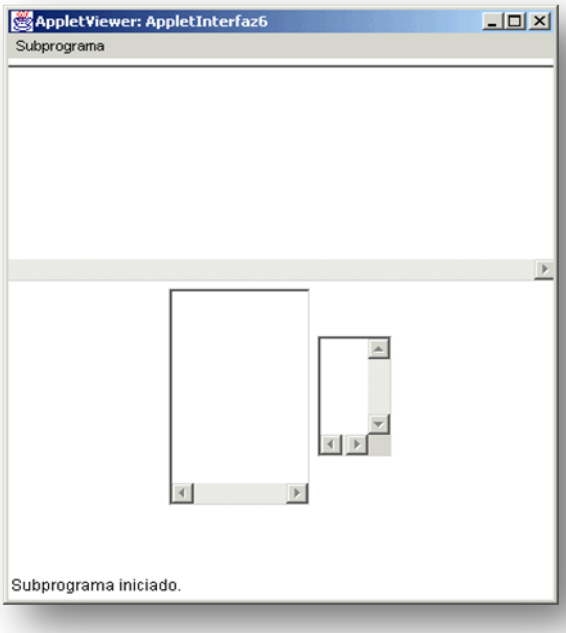
Se deben utilizar con el nombre de la clase como sufijo, ya que son constantes estáticas, ejemplo: TextArea.SCROLLBARS_BOTH.

Ejemplo de aplicación:

```
import java.awt.*;
import java.applet.*;
// <applet width="400" height="400" code="AppletInterfaz6"></applet>
public class AppletInterfaz6 extends Applet {
 TextArea ta1, ta2, ta3;
 public AppletInterfaz6() {
 ta1 = new TextArea();
 ta2 = new TextArea("",10,12,TextArea.SCROLLBARS_HORIZONTAL_ONLY);
 ta3 = new TextArea("",5,5,TextArea.SCROLLBARS_BOTH);
 add(ta1);
 add(ta2);
 add(ta3);
 }
}
```

La manera en la que se visualiza es:

Applet áreas de texto



Fuente: SENA

Existen otros elementos gráficos, por lo que es importante entrar a la biblioteca de clases de la AWT en Java para familiarizarse con ellos.

Tema 2. Administradores de espacio

En Java se hace uso de los administradores de espacio layout manager para poder presentar una buena distribución de los elementos gráficos y no tener necesidad de preocuparse por el tamaño de los botones, textos, entre otros. A continuación se explican los más utilizados:

FlowLayout()

Esta clase es la que se define como administrador de espacio por default al hacer un applet y al añadir un elemento gráfico a la ventana del applet, éste se va creando de izquierda a derecha, de manera que se van acomodando en la ventana.

Cuando se crea un applet se puede definir que este es el layout que se quiere con el método setLayout, ejemplo:

```
setLayout( new FlowLayout());
```

De esta manera es como se van añadiendo los elementos en la ventana.

Todos los ejemplos que se han hecho hasta ahora han utilizado este tipo de administrador de espacio.

GridLayout()

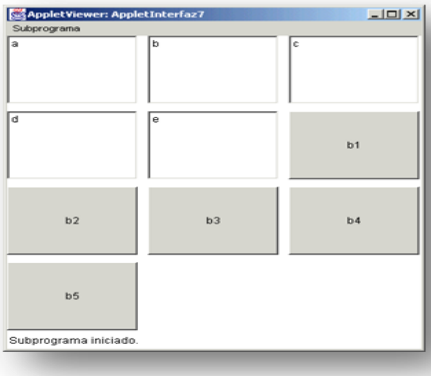
Esta clase permite que se definan elementos en la ventana a manera de renglones y columnas, como si fuera una cuadrícula. Se precisa la cantidad de éstos y los espacios que se quiere dejar en píxeles entre los renglones y las columnas, cada vez que se añade algún elemento este aparece al lado derecho de la hilera en la que va acomodándose, ejemplo:

```
import java.awt.*;
import java.applet.*;
// <applet width="400" height="400" code="AppletInterfaz7"></applet>
public class AppletInterfaz7 extends Applet {
 Button b1, b2, b3, b4, b5;
 TextField t1, t2, t3, t4, t5;
```

```
public AppletInterfaz7() {
setLayout(new GridLayout(4, 3, 10, 10));
t1 = new TextField("a");
t2 = new TextField("b");
t3 = new TextField("c");
t4 = new TextField("d");
t5 = new TextField("e");
b1 = new Button("b1");
b2 = new Button("b2");
b3 = new Button("b3");
b4 = new Button("b4");
b5 = new Button("b5");
add(t1);
add(t2);
add(t3);
add(t4);
add(t5);
add(b1);
add(b2);
add(b3);
add(b4);
add(b5);
}
}
```

El resultado se visualiza así:

Applet GridLayout



Fuente: SENA

De esta manera se van acomodando los elementos de tres en tres por renglón, teniendo cuatro renglones y tres columnas.

BorderLayout()

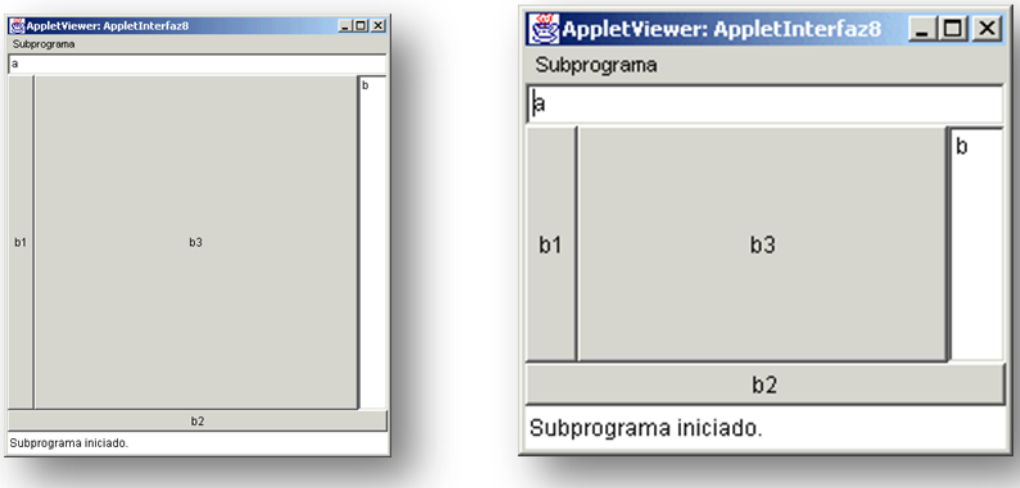
Sirve para poner los elementos por zonas, teniendo el centro, el sur, el norte, el este y el oeste. Cada vez que se añade un elemento se debe decir a qué lugar se manda de la siguiente manera: add(elemento, BorderLayout.NORTH), o en lugar de NORTH puede ser SOUTH, EASTH, WEST o CENTER.

Ejemplo:

```
import java.awt.*;
import java.applet.*;
// <applet width="400" height="400" code="AppletInterfaz8"></applet>
public class AppletInterfaz8 extends Applet {
 Button b1, b2, b3;
 TextField t1, t2;
 public AppletInterfaz8() {
   setLayout(new BorderLayout());
   t1 = new TextField("a");
   t2 = new TextField("b");
   b1 = new Button("b1");
   b2 = new Button("b2");
   b3 = new Button("b3");
   add(t1, BorderLayout.NORTH);
   add(t2, BorderLayout.EAST);
   add(b1, BorderLayout.WEST);
   add(b2, BorderLayout.SOUTH);
   add(b3, BorderLayout.CENTER);
 }
}
```

El resultado se observa:

Applet BorderLayout



Fuente de imágenes: SENA

En la anterior imagen se muestra como al cambiar de tamaño el applet, los elementos gráficos se ajustan para esto, lo cual es una facilidad del layout, GridLayout y del BorderLayout.

Tema 3. Panel

Como ayuda para manejar los elementos gráficos se encuentra el panel, el cual permite tomar elementos gráficos en forma agrupada y manejarlos como si fuera un solo elemento a la hora de añadirlos en una distribución (layout).

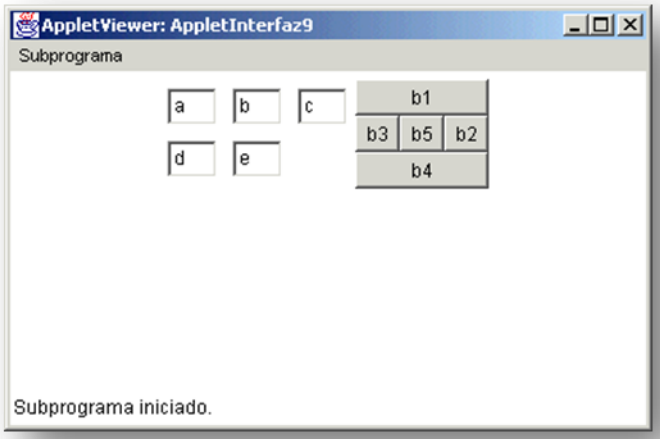
Para crear un panel solamente se define este, e internamente la distribución de los elementos a mostrar, por ejemplo, el panel 1 tiene un GridLayout como distribución y el panel 2 tiene un BorderLayout; éstos pertenecen a un FlowLayout como distribución general, tal y como se muestra a continuación:

```
import java.awt.*;
import java.applet.*;

// <applet width="400" height="200" code="AppletInterfaz9"></applet>
public class AppletInterfaz9 extends Applet {
```

```
        Button b1, b2, b3, b4, b5;
        TextField t1, t2, t3, t4, t5;
        Panel p1, p2;

public AppletInterfaz9() {
                        setLayout(new FlowLayout());
                        p1 = new Panel(new GridLayout(2,2,10,10));
                        p2 = new Panel(new BorderLayout());
                        t1 = new TextField("a");
                        t2 = new TextField("b");
                        t3 = new TextField("c");
                        t4 = new TextField("d");
                        t5 = new TextField("e");
                        b1 = new Button("b1");
                        b2 = new Button("b2");
                        b3 = new Button("b3");
                        b4 = new Button("b4");
                        b5 = new Button("b5");
                        p1.add(t1);
                        p1.add(t2);
                        p1.add(t3);
                        p1.add(t4);
                        p1.add(t5);
                        p2.add(b1,BorderLayout.NORTH);
                        p2.add(b2, BorderLayout.EAST);
                        p2.add(b3, BorderLayout.WEST);
                        p2.add(b4, BorderLayout.SOUTH);
                        p2.add(b5, BorderLayout.CENTER);
                        add(p1);
                        add(p2);

        }
}
```

El resultado se visualiza así:

Applet paneles



Fuente: SENA

Una vez definidos los paneles y añadidos los elementos, es importante no olvidar agregar los paneles al applet al final, por eso se tiene add(p1) y add(p2) al final.

Los paneles se pueden utilizar como desee, cada uno de ellos puede ser tratado como la ventana del applet, de manera que se defina un diferente manejador de espacio, tal y como se observó anteriormente.

Inclusive se puede tener un panel cuyo manejador de espacio es un GridLayout, pero hacia adentro; uno de los elementos que se le agrega a ese panel puede ser otro panel que contenga elementos añadidos con un manejador de espacio de FlowLayout o BorderLayout. En fin se puede utilizar la imaginación para definir las diferentes presentaciones del applet.

Hasta se puede manejar un panel dentro de otro, teniendo cuidado de utilizar el método add correctamente en el panel adecuado. Cuando se usa p1.add(objeto), lo que se hace es añadir el objeto gráfico al panel p1, pero esa instrucción aún no inserta algo al applet, por eso se debe tener cuidado de agregar finalmente al applet, el panel o los paneles finales, es decir los que ya contengan la información deseada con los correspondientes objetos creados.

Referencias

- Sánchez, M. (s.f.). *Gestión de eventos*. Consultado el 24 de abril 2014, en http://odelys2003.files.wordpress.com/2013/01/1-1-javagestioneventos-100310084758-phpapp01.pdf

- Servicio Nacional de Aprendizaje, SENA. (2010). *Desarrollo de aplicaciones con interfaz gráfica, manejo de eventos, clases y objetos: Java.* Colombia: Autor.

Control del documento

	Nombre	Cargo	Dependencia	Fecha
Autor	Jorge Hernán Moreno Tenjo	Instructor	Centro Metalmecánico. Regional Distrito Capital	Mayo de 2013
Adaptación	Ana María Mora Jaramillo	Guionista - Línea de producción	Centro Agroindustrial. Regional Quindío	Mayo de 2014
	Rachman Bustillo Martínez	Guionista - Línea de Producción	Centro Agroindustrial. Regional Quindío	Mayo de 2014