élogos

CREACIÓN DE VENTANAS Y COMPONENTES

élogos

ÍNDICE

CREACIÓN DE VENTANAS Y COMPONENTES

Creación de ventanas y componentes

Objetivos

Al finalizar esta lección serás capaz de:

- Utilizar las clases Frame y JFrame para crear ventanas
- Añadir componentes gráficos a una ventana.
- Conocer los principales controles que se utilizan en una interfaz gráfica

1. Creación de una ventana

La clase Frame

La clase Frame del paquete java.awt proporciona el soporte básico para la creación de ventanas en AWT. A través del constructor Frame(String title) se puede crear una ventana vacía con un determinado título.

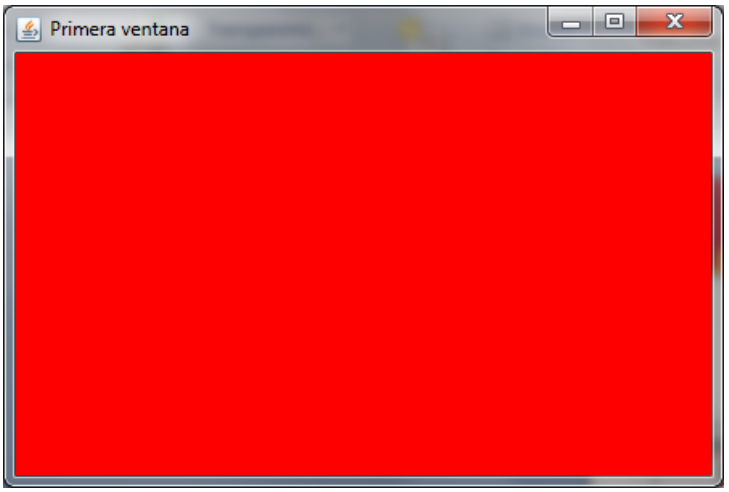
Una vez creado el objeto, podemos utilizar los siguientes métodos heredados de Component para jugar con el aspecto de la ventana:

- **setBounds:** Establece la posición y tamaño de la ventana. Los valores x e y determinan las coordenadas de la esquina superior izquierda respecto a la pantalla, mientras que w y h son el ancho y alto de la ventana respectivamente. Todo ello medido en pixels.

- **setBackground:** Establece el color de fondo de la ventana. El color se define a través de la clase Color, también del AWT.

- **setVisible:** Para que la ventana se muestre, es necesario visualizarla llamando a este método con el valor true.

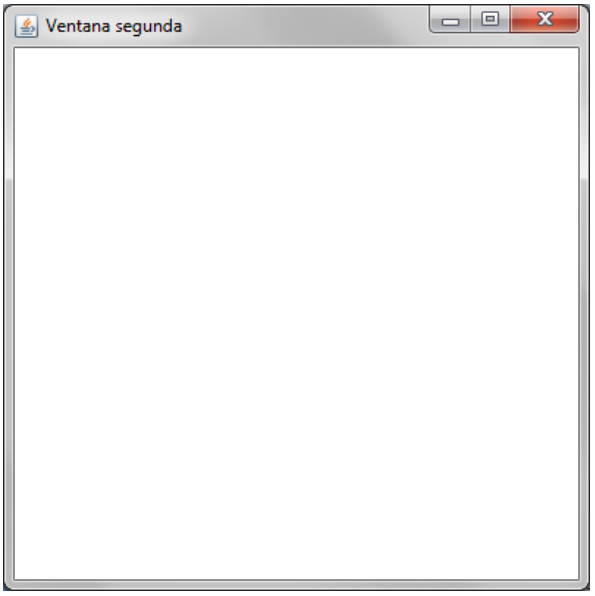
```
public static void main(String[] args) {
        Frame f=new Frame("Primera ventana");
        f.setBounds(100,50,500,300);
        f.setBackground(Color.red);
        f.setVisible(true);
    }
```



Herencia de la clase Frame

Normalmente, a la hora de trabajar con ventanas se define una clase que herede Frame, indicando en el constructor de la nueva clase las instrucciones necesarias para la configuración de la ventana.

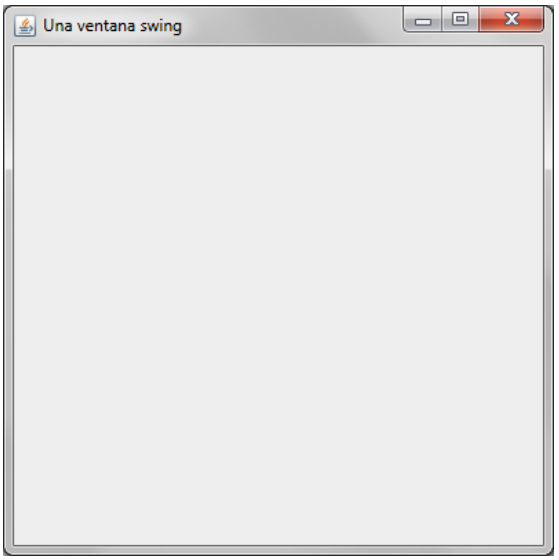
```
import java.awt.*;
public class Ventana extends Frame{
    public Ventana(String titulo, int x, int y, int w, int h){
        super(titulo);
        this.setBounds(x, y, h, h);
        this.setVisible(true);
    }
    public Ventana(){
        //valores de creación por defecto
        this("ventana por defecto",20,20,400,300);
    }
    public static void main(String[] args){
        //crea el objeto Ventana
        new Ventana("Ventana segunda", 100, 50, 600,400);
    }
}
```



La clase JFrame

La creación de ventanas en swing sigue el mismo esquema que en AWT, solo que en esta ocasión empleamos la clase JFrame, que es una subclase de Frame:

```
import javax.swing.*;
public class JVentana extends JFrame{
    public JVentana(String titulo, int x, int y, int w, int h){
        super(titulo);
        this.setBounds(x, y, h, h);
        this.setVisible(true);
    }
    public JVentana(){
        //valores de creación por defecto
        this("ventana por defecto",20,20,400,300);
    }
    public static void main(String[] args){
        //crea el objeto JVentana
        new JVentana("Una ventana swing", 100, 50, 600,400);
    }
}
```



Cierre de una ventana

La pulsación del botón de cierre en una ventana Frame no provoca ningún efecto; sería necesario programar las acciones de cierre de la ventana en el método de respuesta al evento.

En JFrame existe la posibilidad de definir el comportamiento de la ventana cuando el botón de cierre sea pulsado. Dicho comportamiento se establece a través del método setDefaultCloseOperation (int opt). Los posibles valores que se le pueden pasar a este método están definidos en las siguientes constantes:

- **DO_NOTHING_ON_CLOSE**. No sucede nada al pulsar el botón. Definida en la interfaz WindowConstants.
- **HIDE_ON_CLOSE**. La ventana se oculta, pero sigue cargada en memoria. Definida en la interfaz WindowConstants.

- **DISPOSE_ON_CLOSE**. La ventana se cierra y, por tanto, se descarga de memoria. Definida en la interfaz WindowConstants.

- **EXIT_ON_CLOSE**. El programa finaliza. Definida en Jframe.

2. Otros contenedores

Cuadros de diálogo

Un cuadro de diálogo es similar a una ventana, con la diferencia de que puede trabajar de forma modal, lo que significa que se puede impedir que el usuario interactúe con el resto de la aplicación mientras el cuadro está abierto.

Tanto AWT como swing disponen clases para creación de cuadros de diálogo, concretamente, Dialog y Jdialog, ambas son muy similares.

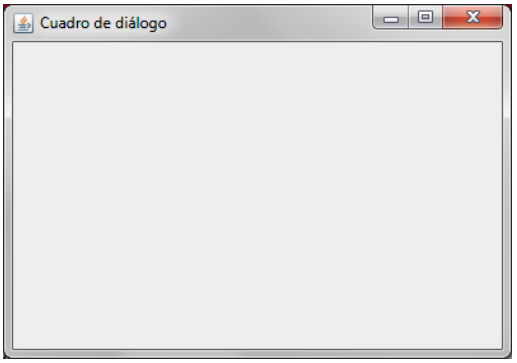
Los constructores más utilizados de estas dos clases serían:

- Dialog (Frame f, boolean modal)
- JDialog (Frame f, boolean modal)

En ambos tipos de componentes se requiere un objeto Frame que sirva de base para crear el cuadro de diálogo. El parámetro boolean permite activar la opción modal (true).

```
public class JDialogo extends JDialog{
   public JDialogo(String titulo, int x, int y, int w, int h){
      super(new JFrame(titulo), true);
      this.setBounds(x, y, h, h);
      this.setVisible(true);
   }
   public JDialogo(){
      //valores de creación por defecto
      this("cuadro por defecto",20,20,400,300);
   }
```

```
    public static void main(String[] args){
        //crea el objeto JVentana
        new JVentana("Cuadro de diálogo", 100, 50, 600,400);
    }
}
```



Applets

Aunque se analizarán con más detalle en una lección posterior, un applet es una aplicación que se ejecuta dentro de un navegador. La clase base para la creación de applets es Applet o JApplet.

La implementación es:

```
import javax.swing.JApplet;
public class MiAppletextends  JApplet {
 public void init(){
     //instrucciones para creación de componentes
     //de la interfaz
 }
}
```

La clase JApplet hereda a su vez de Jpanel. El aspecto de un applet en un navegador es el de un panel, es decir, un contenedor sin barra de título y sin borde.

3. Controles AWT

Añadir controles al contenedor

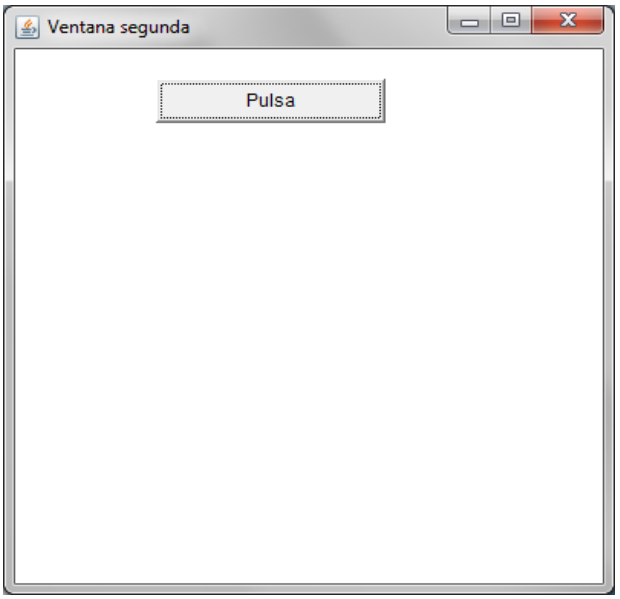
Independientemente del control que se trate, se deberá añadir al contenedor una vez creado. Normalmente, este proceso se realiza en el constructor del contenedor o método init() en el caso de los applets. Los pasos a seguir para añadir los componentes al contenedor son sencillos:

Creamos el componente. Como se ha indicado, en el constructor o init().

Añadimos el componente al contenedor a través del método add() heredado de Container.

Le asignamos una localización y tamaño. Aunque para que sea tenida en cuenta, habrá que desactivar el gestor de organización del contenedor

```
public class Ventana extends Frame{

   public Ventana(String titulo, int x, int y, int w, int h){

      super(titulo);

      //colocaciónventana

      this.setBounds(x, y, h, h);

      //eliminación gestor organización

      this.setLayout(null);

      //controles

      Button b=new Button("Pulsa");

      b.setBounds(100, 50, 150, 30);

      this.add(b);

      //visualizar ventana
```



```
    this.setVisible(true);
  }
:
}
```

Etiquetas

Las Etiquetas son cadenas de texto que sirven para nombrar o etiquetar los otros componentes. La ventaja respecto al texto propiamente es que no tenemos que repintarlas nosotros y además se alinean en el panel.

Para crear una etiqueta usaremos uno de los siguientes constructores:

Label ()

Crea una etiqueta vacía, su texto se alinea a la izquierda

Label(String)

Crea una etiqueta con el string que le pasamos alineado a la izquierda.

Label(String, int)

Crea una etiqueta con el string que le pasamos y la alinea según el valor que le pasamos. Los alineamientos posibles son 3. Así la variable int será: Label.RIGHT, Label.LEFT o Label.CENTER

La fuente que escribe el contenido de la etiqueta depende de la fuente que hayamos configurado con setFont().

Ejemplo:

```
add( new Label("Texto centrado", Label.CENTER));
```

Una vez definida una etiqueta podemos utilizar métodos predefinidos en la clase Label.

Método	Acción
getText()	Devuelve un string con el contenido de esta .
setText(String)	Cambia el texto de la etiqueta a este.
getAlignment()	Devuelve el alineamiento: 0 si Label.LEFT 1 si Label.CENTER 2 si Label.RIGHT
setAlignment(int)	Cambia el alineamiento según el int dado.

Botones

Botones son los elementos que sirven para realizar una acción cuando son pulsados.

Para crear un botón usaremos uno de estos constructores:

- **Button ():** crea un botón vacío.
- **Button (String):** crea un botón con el string como etiqueta.

Una vez creado el botón podemos usar estos métodos con él.

Método	Acción
getLabel()	devuelve su etiqueta.
setLabel(string)	asignamos una nueva etiqueta al botón.

El tamaño del botón lo da la fuente de texto que lo etiqueta.

Checkboxes

Los Checkboxes son interfaces de usuario que pueden estar activadas o desactivadas. Son pequeñas cajas cuadradas (casillas de verificación) que se pueden marcar o desmarcar.

Las checkboxes pueden ser utilizadas de dos formas:

- **Checkboxes no exclusivas:** Cualquiera de un grupo puede estar seleccionada o no.
- **Checkboxes exclusivas:** Una sola checkbox de un grupo puede estar activada. Se llaman radio buttons o también checkbox groups.

En este apartado veremos solo las **checkboxes** no exclusivas. Para su creación utilizaremos uno de los siguientes constructores:

- **Checkbox():** crea una checkbox vacía.
- **Checkbox(String):** crea una checkbox con la etiqueta que le pasamos.
- **Checkbox(String, null, boolean):** crea la checkbox con la etiqueta que le pasamos y que estará activada/desactivada si el boolean vale true/false. El valor null es característico de estos componentes.

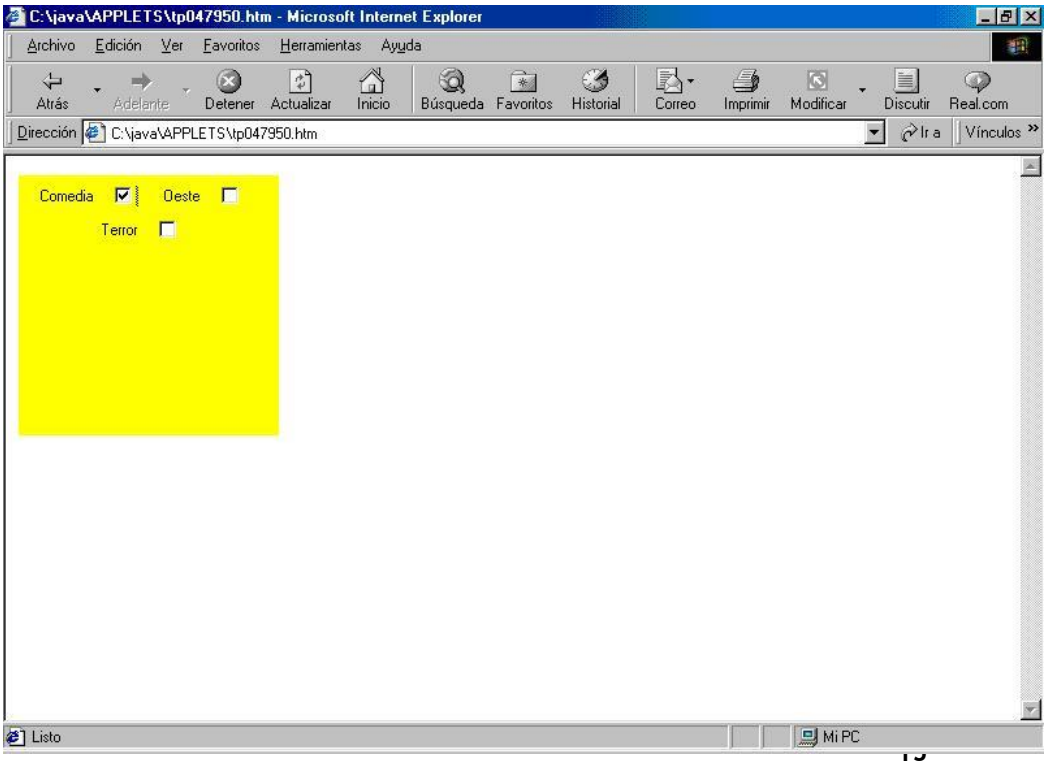
Método	Acción
getLabel()	devuelve su etiqueta.
setLabel(string)	asignamos una nueva etiqueta al botón.
getState()	devuelve true/false según este o no activada.
setState(boolean)	fija el estado al boolean que le pasemos.

```
import java.awt.*;

import java.applet.Applet;

public class Cajas extends Applet

{

Label l1=new Label("Comedia");

Label l2=new Label("Oeste");
```

```
Label l3=new Label("Terror");
Checkbox c1=new Checkbox();
Checkbox c2=new Checkbox();
Checkbox c3=new Checkbox();
public void init()
{
setBackground(Color.yellow);
add(l1);
add(c1);
add(l2);
add(c2);
add(l3);
add(c3);
}
}
```

Código HTML:

```
<html>
<body>
<applet
code="Cajas.class"
width=200
height=200>
```



```
</applet>

</body>

</html>
```

RadioButtons

RadioButton son un grupo de checkboxes exclusivas. Para crearlos utilizaremos el siguiente constructor mediante un ejemplo.

```
CheckboxGroup cbg = new CheckboxGroup();

add(new Checkbox("Rojo",cbg,true));

add(new Checkbox("Azul",cbg,false));

add(new Checkbox("Verde",cbg,false));

add(new Checkbox("Amarillo",cbg,false));
```

Además podemos utilizar otros métodos predefinidos para los checkbox componentes del radio button:

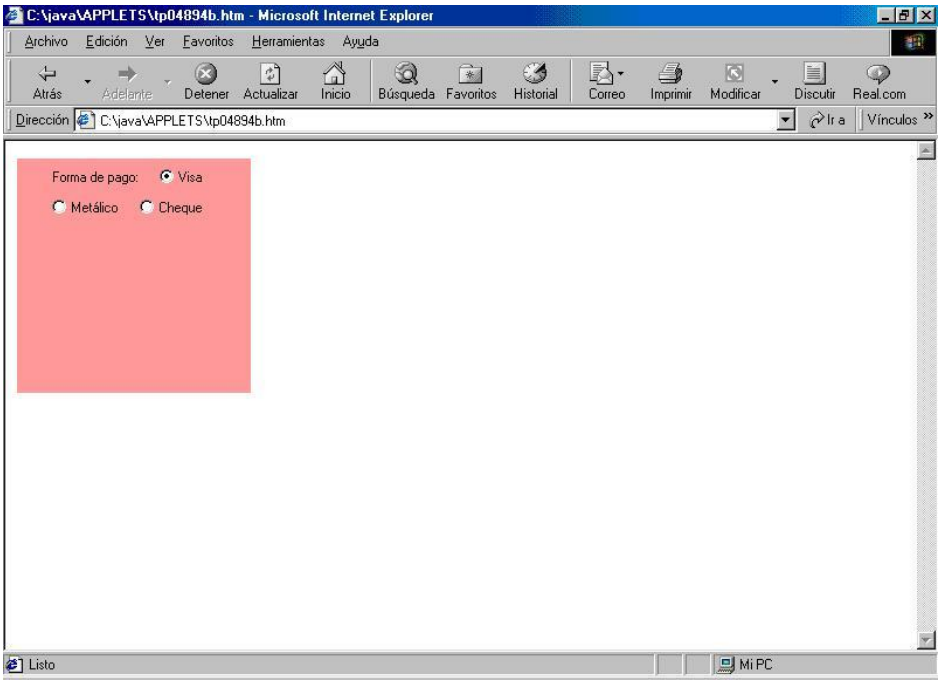
Método	Acción
getCheckboxGroup()	conocer el grupo del checkbox.
setCheckboxGroup()	cambiar el grupo de un checkbox.
getCurrent()	saber que checkbox es el activado.
setCurrent(Checkbox)	para activar un checkbox en concreto.

```
import java.awt.*;

import java.applet.Applet;

public class RadiosButons extends Applet

{

Label l1=new Label("Forma de pago:");

public void init()
```

```
{
setBackground (Color.pink);
CheckboxGroup cbg = new CheckboxGroup();
add(l1);
add(new Checkbox("Visa",cbg,true));
add(new Checkbox("Metálico",cbg,false));
add(new Checkbox("Cheque",cbg,false));
}
}
```

Código HTML:

```
<html>
<body>
<applet
code="RadiosButons.class“
width=200
height=200>
</applet>
</body>
</html>
```



Menús de selección

Los menús de selección son menús desplegables que permiten seleccionar un ítem. Después el menú representa ese ítem en la pantalla.

Para crear un **choice menu** creamos un objeto de la clase **Choice**, luego añadimos los ítems y finalmente incluimos el menú en el **panel**:

```
Choice c = new Choice();

c.addItem("Manzanas");

c.addItem("Naranjas");

c.addItem("Platanos");

c.addItem("Melocotones");

add(c);
```

Estos menús solo permiten una selección por menú.

Más **métodos** para este componente son:

Método	Acción
addItem (String)	añade el nuevo ítem con su etiqueta.
getitem (int)	devuelve el string del ítem indicado. [0,n-1]
countItems()	devuelve el numero de ítems del menú.
getSelectedIndex()	devuelve la posición del ítem seleccionado.
getSelectedItem ()	devuelve el ítem seleccionado como string.
select(int)	selecciona el ítem que le pasamos.
select(String)	selecciona el ítem con el string pasado.

Campos de texto

Los campos de texto son componentes que nos permiten incluir valores en forma de texto.

Para crear un campo de texto, el **constructor** toma varias formas.

Método	Acción
TextField()	crea un campo de texto vacío de 0 caracteres de anchura
TextField(int)	crea un campo de texto vacío con el numero de caracteres que le pasamos
TextField(String)	crea el texto de tamaño 0, inicializado con el string que le pasamos
TextField(String, int)	crea el campo de texto con el string pasado y el tamaño int que le damos

Después de crear el campo de texto lo sumamos al **panel**.

Ejemplo: **TextField tf = new TextField("Hola desde Java",30);**

add(tf);

El campo de texto solo incluye el campo editable y no incluye ninguna etiqueta. Seremos nosotros los que incluyamos la nueva etiqueta.

Los campos de texto sólo ocupan una línea y están limitados a diferencia de las áreas de texto que hasta pueden tener barras de scroll.

Además podemos crear el campo de texto típico de las contraseñas que esconda el texto. Para ello crearemos el campo de texto como tal y después teclearemos el **setEchoCharacter('*')**.

Ejemplo: **TextField tf = new TextField("Escribe algo",30);**

tf.setEchoCharacter('*');

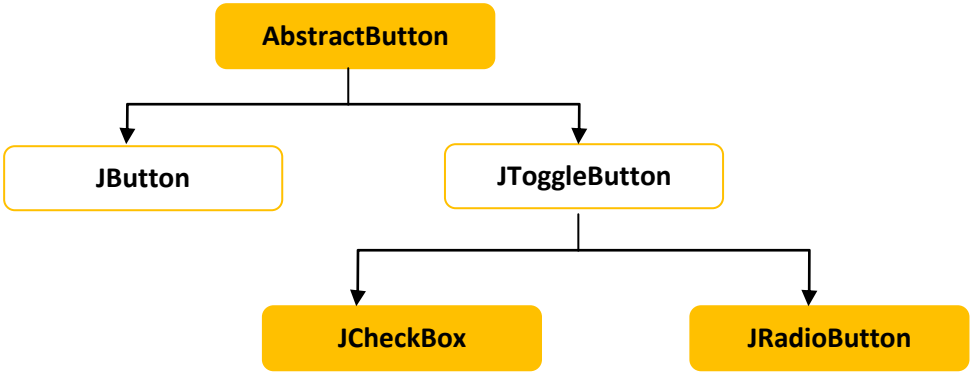
add(tf);

Creación de ventanas y componentes

Método	Acción
getText()	devuelve el contenido como un string
setText(String)	pone el texto que pasamos dentro del campo
getColumns()	devuelve la anchura del campo de texto
select(int,int)	selecciona el texto entre las dos posiciones dadas como argumentos de entrada
selectAll()	selecciona todo el texto en el campo
isEditable()	devuelve true/false segun el texto sea editable o no
setEditable(boolean)	true hace que el texto sea editable, false hace lo contrario
getEchoChar()	devuelve el carácter utilizado como mascara
echoCharIsSet()	devuelve true o false según el campo tenga un carácter mascara o no

4. Controles swing

Los botones



void <AbstractButton> setEnabled(boolean b)

Activarlo o desactivarlo

Todos los botones tienen, además del texto, un Icono y un Mnemonic (tecla rápida).

```
void <AbstractButton>.setText(String text)
String <AbstractButton>.getText()
void <AbstractButton>.setIcon(Icon icon)
Icon <AbstractButton>.getIcon()
void <AbstractButton>.setMnemonic(char c)
char <AbstractButton>.getMnemonic()
```

JButton →Botón pinchable

JChekBox →CheckBox

JRadioButton →Radio Button

JToggleButton → Un botón que cuando se pincha se aplasta y cuando se vuelve a pinchar se levanta.

JButton

Los constructores de JButton son los siguientes:

- **JButton()**
- **JButton(String text)**
- **JButton(Icon icon)**
- **JButton(String text, Icon icon)**

Se le indica al formulario cuál es el botón por defecto: **void <JRootPane> setDefaultButton(JButton boton)**

Para indicar la posición del texto respecto al icono: **void <AbstractButton> setVerticalTextPosition(int pos)**

AbstractButton.CENTER (defecto)

AbstractButton.TOP

AbstractButton.BOTTOM

void <AbstractButton>.setHorizontalTextPosition(int pos)

AbstractButton.LEFT

AbstractButton.CENTER

AbstractButton.RIGHT (defecto)

JCheckBox

Los constructores de JcheckBox son:

JCheckBox()

JCheckBox(String text)

JCheckBox(String text,Icon icon)

Eventos: **ItemEvent**

ActionEvent

Se produce cada vez que pulsamos en el **CheckBox**. El más utilizado es el **ActionEvent**, ya que se utiliza junto al método:

int <ItemEvent>.getStateChange()

Devuelve el estado del CheckBox. Se puede saber lo que ha ocurrido.

JRadioButton

Existe una clase para agruparlos: **ButtonGroup (derivada de Object).**

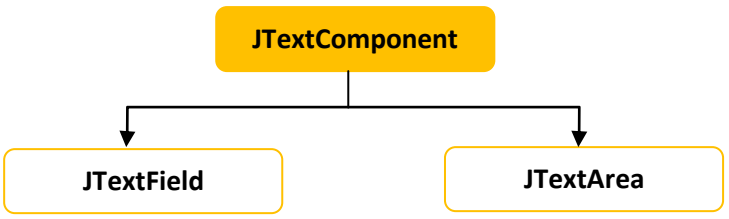
void <ButtonGroup> add (AbstractButton b)

Se añadirán los RadioButton.

```
JRadioButton rojo  = new JRadioButton("rojo");
JRadioButton verde = new
JRadioButton("verde",true);
JRadioButton azul  = new JRadioButton("azul");
ButtonGroup grupo  = new ButtonGroup();
grupo.add(rojo);
grupo.add(verde);
grupo.add(azul);
```

Los campos de texto

Las cajas de texto son subclases de JTextComponent:



JTextField → Caja de texto única línea. JTextField(String textoini)

JTextArea → Caja de texto multilinea. JTextArea(String textoini)

Entre los principales métodos de JTextComponent están:

```
void setText(String text)

String getText()

String getSelectedText()

void setEditable(boolean b)
```