

CURSO INICIACIÓN JAVA IV

Tutoriales de pildorasinformaticas

Descripción breve

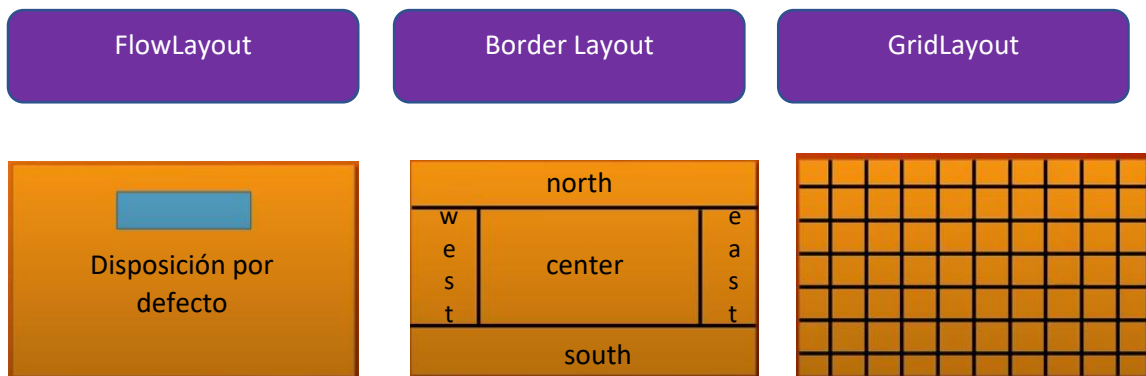
Curso introductorio de Java por pildorasinformaticas.



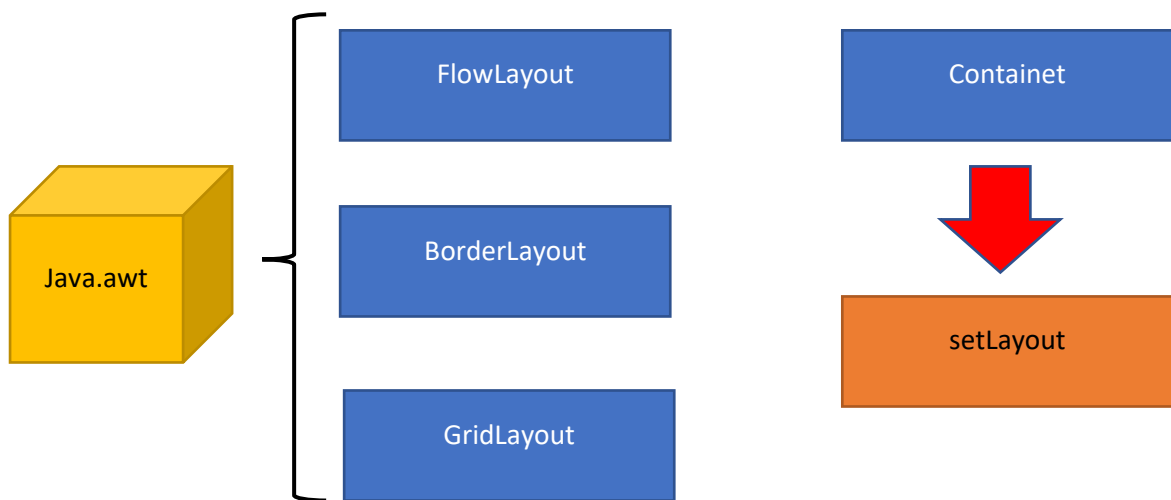
Pere Manel Verdugo Zamora
pereverdugo@gmail.com

Layouts I (VÍdeo 81)

Layouts (Disposiciones)



Si ninguna de estas disposiciones te va bien puedes crear las tuyas propias.



¿Cómo se trabaja con FlowLayout?

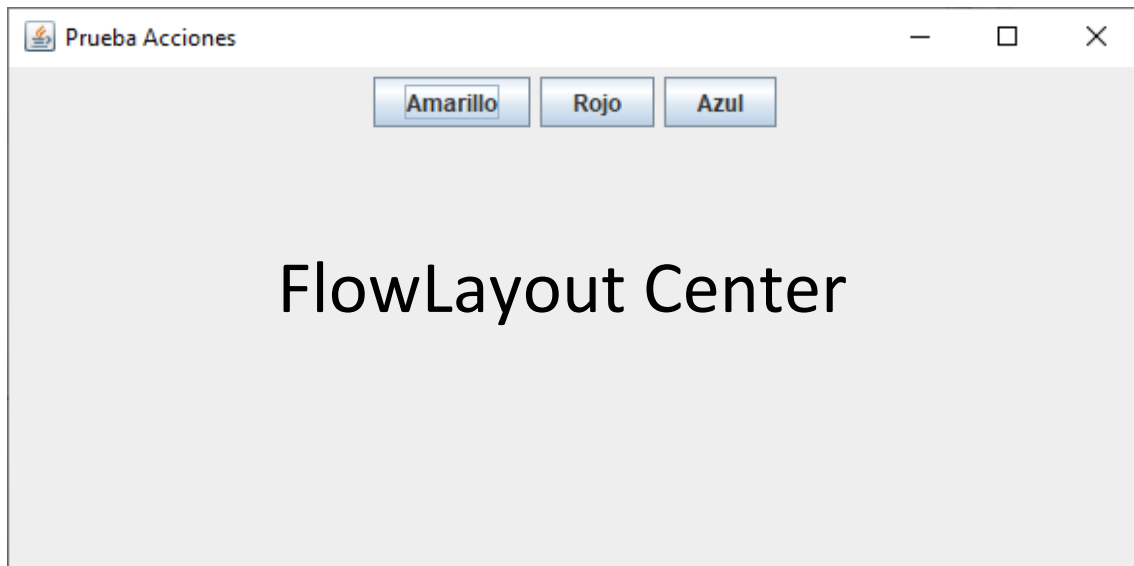
```
1 package graficos;
2 import java.awt.*;
3 import javax.swing.*;
4
5 public class Layouts_I {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         Marco_Layout marco=new Marco_Layout();
10        marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        marco.setVisible(true);
12    }
13 }
```

```

14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         add(lamina);
20     }
21 }
22 class Panel_Layout extends JPanel{
23     public Panel_Layout() {
24         add(new JButton("Amarillo"));
25         add(new JButton("Rojo"));
26         add(new JButton("Azul"));
27     }
28 }
29 }
30 }

```

Este será el resultado:



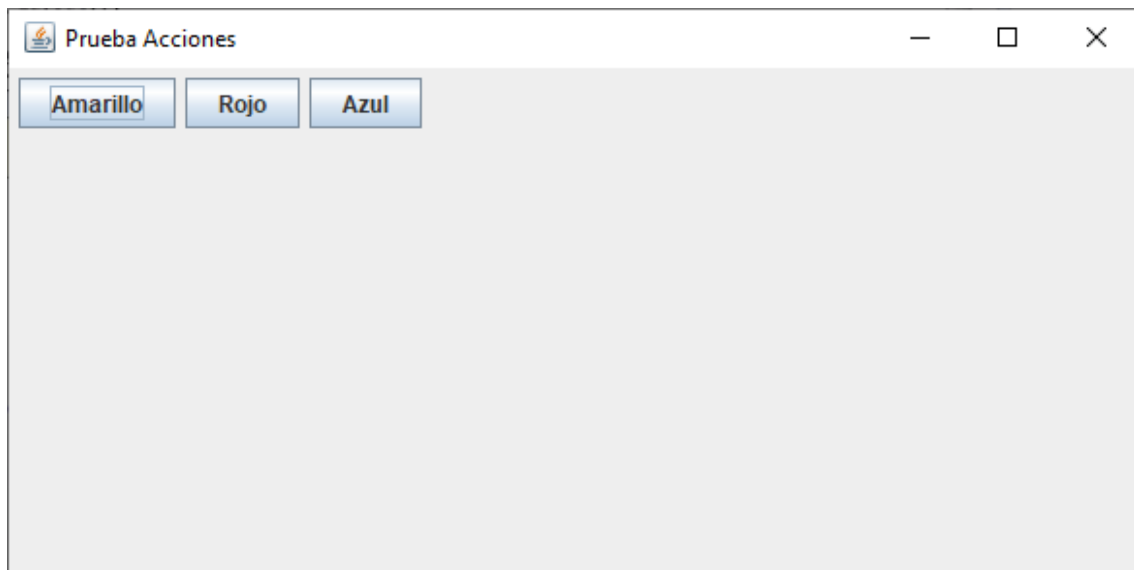
Layout puede ser Center (por defecto), left y right y una separación de 5 pixeles.

```

14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         FlowLayout diposicion = new FlowLayout(FlowLayout.LEFT);
20         lamina.setLayout(diposicion);
21         add(lamina);
22     }
23 }

```

Para linear los botones hacia la izquierda, si ejecutamos este será el resultado:



```
14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         FlowLayout diposicion = new FlowLayout(FlowLayout.RIGHT);
20         lamina.setLayout(diposicion);
21         add(lamina);
22     }
23 }
```

Si cambiamos por RIGHT la alineación será a la derecha.



Si ponemos CENTER es lo mismo que si no ponemos nada, es por defecto.

Una manera de simplificar las líneas 19 y 20.

```
14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         //FlowLayout diposicion = new FlowLayout(FlowLayout.RIGHT);
20         //lamina.setLayout(diposicion);
21         lamina.setLayout(new FlowLayout(FlowLayout.LEFT));|
22         add(lamina);
23     }
24 }
```

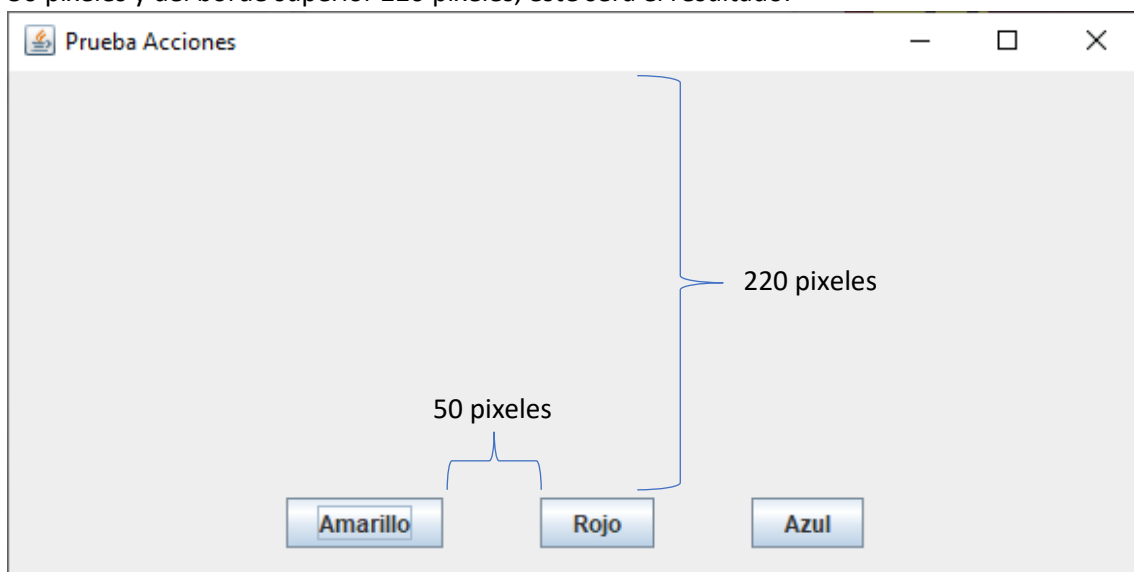
La línea 21 sustituye a las líneas 19 y 20.



Layouts II. (VÍdeo 82)

```
14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         //FlowLayout diposicion = new FlowLayout(FlowLayout.RIGHT);
20         //lamina.setLayout(diposicion);
21         lamina.setLayout(new FlowLayout(FlowLayout.CENTER,50,220));
22         add(lamina);
23     }
24 }
```

Si agregamos dos parámetros más le estamos diciendo que separe cada botón entre ellos de 50 píxeles y del borde superior 220 píxeles, este será el resultado.



Se puede borrar de la clase Marco_Layout y agregarlo a la clase Panel_Layout, funciona igualmente.

```
25 class Panel_Layout extends JPanel{
26     public Panel_Layout() {
27         setLayout(new FlowLayout(FlowLayout.CENTER,50,220));
28         add(new JButton("Amarillo"));
29         add(new JButton("Rojo"));
30         add(new JButton("Azul"));
31     }
32 }
```

Ahora vamos a trabajar con el siguiente distanciamiento BorderLayout.

```
25 class Panel_Layout extends JPanel{
26     public Panel_Layout() {
27         setLayout(new BorderLayout());
28         add(new JButton("Amarillo"), BorderLayout.NORTH);
29         add(new JButton("Rojo"), BorderLayout.SOUTH);
30         add(new JButton("Azul"), BorderLayout.WEST);
31         add(new JButton("Verde"), BorderLayout.EAST);
32         add(new JButton("Negro"), BorderLayout.CENTER);
33     }
34 }
```

Este será el resultado:



Hay otro constructor BorderLayout(int hgap, int vgap) que admite dos parámetros.

```

25 class Panel_Layout extends JPanel{
26     public Panel_Layout() {
27         setLayout(new BorderLayout(10,10));
28         add(new JButton("Amarillo"), BorderLayout.NORTH);
29         add(new JButton("Rojo"), BorderLayout.SOUTH);
30         add(new JButton("Azul"), BorderLayout.WEST);
31         add(new JButton("Verde"), BorderLayout.EAST);
32         add(new JButton("Negro"), BorderLayout.CENTER);
33     }
34 }

```

Este será el resultado:



En el siguiente ejemplo vamos a ver como combinar dos disposiciones de marco, FloatLayout y BorderLayout.

Vamos a crear dos clases con distinta disposición de botones numerada 1 y 2.

En la clase Marco_Layout definimos las dos laminas, numerada 3.

Y a continuación las agregamos en distinta disposición, si no es así una lámina sobrescribiría a la otra, numerada 4.

```
14 class Marco_Layout extends JFrame{
15     public Marco_Layout() {
16         setTitle("Prueba Acciones");
17         setBounds(600,350,600,300);
18         Panel_Layout lamina=new Panel_Layout();
19         Panel_Layout2 lamina2=new Panel_Layout2();
20
21         add(lamina,BorderLayout.NORTH);
22         add(lamina2, BorderLayout.SOUTH);
23     }
24     class Panel_Layout extends JPanel{
25     public Panel_Layout() {
26         setLayout(new FlowLayout(FlowLayout.LEFT));
27         add(new JButton("Amarillo"));
28         add(new JButton("Rojo"));
29     }
30     }
31 }
32
33 class Panel_Layout2 extends JPanel {
34     public Panel_Layout2() {
35         setLayout(new BorderLayout());
36         add(new JButton("Azul"), BorderLayout.WEST);
37         add(new JButton("Verde"),BorderLayout.EAST);
38         add(new JButton("Negro"),BorderLayout.CENTER);
39     }
40 }
41 }
```

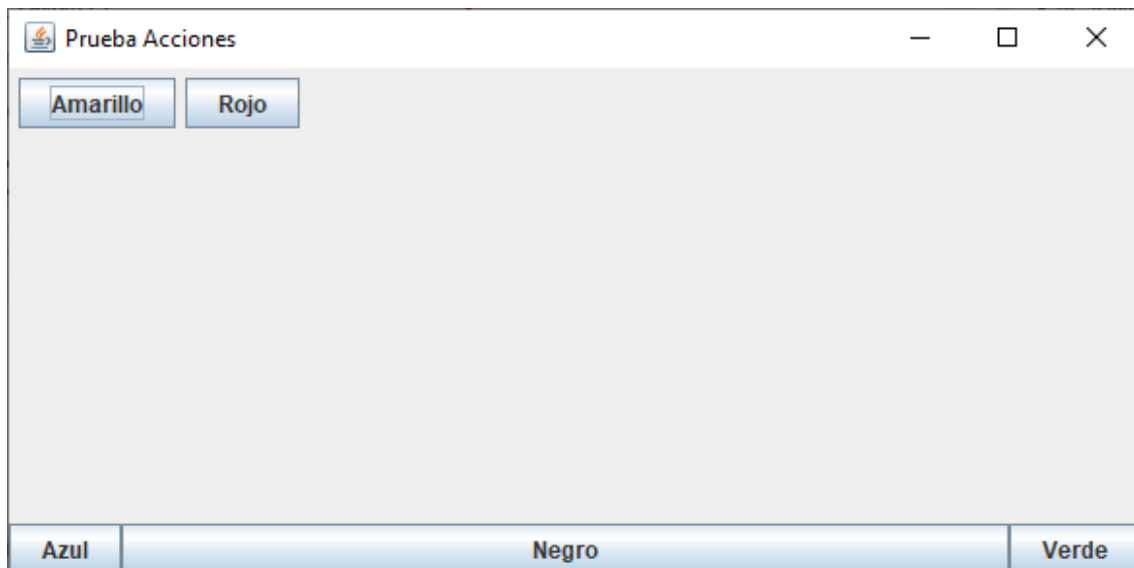
3

4

1

2

Este será el resultado:



Ahora modificando el código que te quede de la siguiente forma:



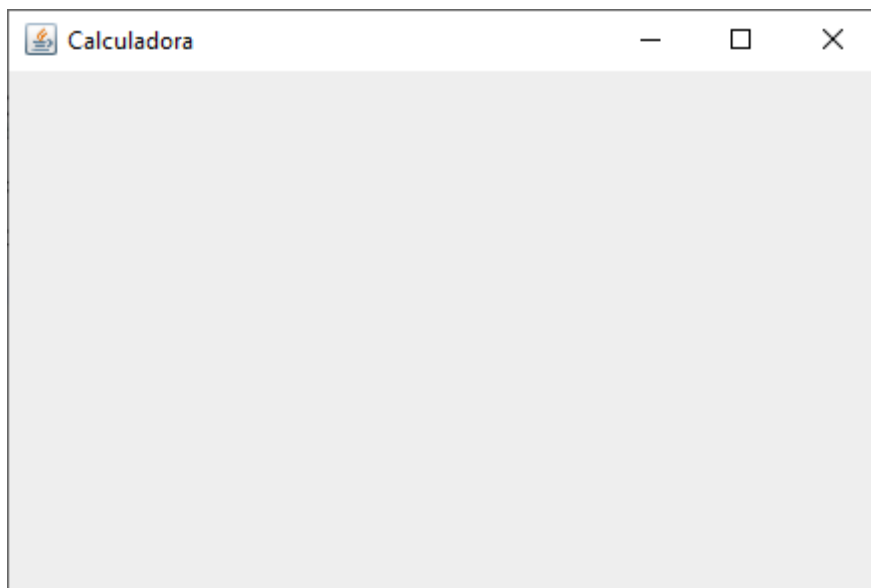
Layouts III. (VÍdeo 83)

En este capítulo vamos a ver la disposición GridLayout.

Vamos a crear una clase nueva llamada Calculadora.

```
1 package graficos;
2
3 import javax.swing.*;
4
5 public class Calculadora {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoCalculadora mimarco=new MarcoCalculadora();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        mimarco.setVisible(true);
12    }
13 }
14 class MarcoCalculadora extends JFrame{
15     public MarcoCalculadora() {
16         setTitle("Calculadora");
17         setBounds(50,300,450,300);
18     }
19 }
20
```

Solo hemos construido la ventana:



```

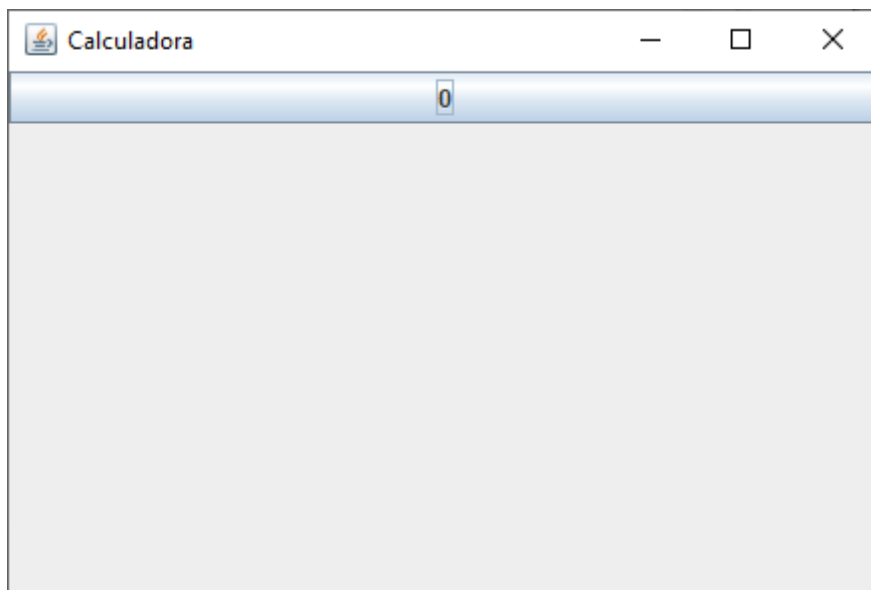
1 package graficos;
2
3 import java.awt.*; ←
4
5 import javax.swing.*;
6
7 public class Calculadora {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         MarcoCalculadora mimarco=new MarcoCalculadora();
12         mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         mimarco.setVisible(true);
14     }
15 }
16 class MarcoCalculadora extends JFrame{
17     public MarcoCalculadora() {
18         setTitle("Calculadora");
19         setBounds(50,300,450,300);
20         LaminaCalculadora milamina=new LaminaCalculadora();
21         add(milamina);
22     }
23 }
24 class LaminaCalculadora extends JPanel{
25     public LaminaCalculadora() {
26         setLayout(new BorderLayout());
27         JButton pantalla =new JButton("0");
28         add(pantalla, BorderLayout.NORTH);
29     }
30 }

```

Importamos java.awt.*;

Creamos la clase LaminaCalculadora , donde le decimos el tipo de disposición de un botón con el título 0, este se sitúa en la parte norte.

En la clase MarcoCalculadora definimos un objeto de tipo LaminaCalculadora llamado milamina que a continuación la agregamos.



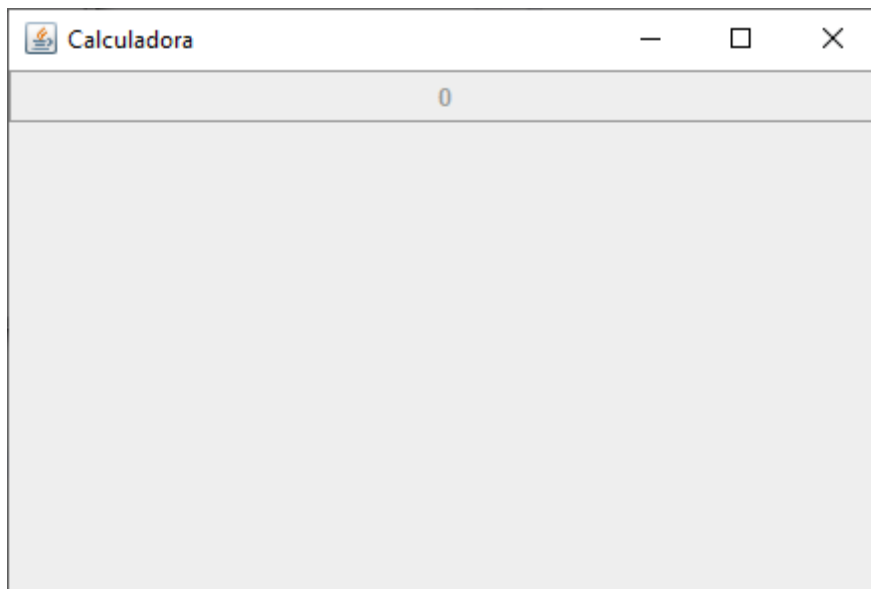
A continuación vamos a rehabilitar el botón.

```

class LaminaCalculadora extends JPanel{
    public LaminaCalculadora() {
        setLayout(new BorderLayout());
        JButton pantalla =new JButton("0");
        pantalla.setEnabled(false); ←
        add(pantalla, BorderLayout.NORTH);
    }
}

```

Este será el resultado:



Ahora vamos a introducir el código de la clase LaminaCalculadora:

```

class LaminaCalculadora extends JPanel{
    public LaminaCalculadora() {
        setLayout(new BorderLayout());
        JButton pantalla =new JButton("0");
        pantalla.setEnabled(false);
        add(pantalla, BorderLayout.NORTH);
        JPanel milamina2=new JPanel();
        milamina2.setLayout(new GridLayout(4,4));

        JButton boton1=new JButton("1");
        milamina2.add(boton1);
        JButton boton2=new JButton("2");
        milamina2.add(boton2);
        JButton boton3=new JButton("3");
        milamina2.add(boton3);
        JButton botonSuma=new JButton("+");
        milamina2.add(botonSuma);

        JButton boton4=new JButton("4");
        milamina2.add(boton4);
        JButton boton5=new JButton("5");
        milamina2.add(boton5);
        JButton boton6=new JButton("6");
        milamina2.add(boton6);
        JButton botonResta=new JButton("-");
        milamina2.add(botonResta);
    }
}

```

```

        JButton boton7=new JButton("7");
        milamina2.add(boton7);
        JButton boton8=new JButton("8");
        milamina2.add(boton8);
        JButton boton9=new JButton("9");
        milamina2.add(boton9);
        JButton botonMultiplica=new JButton("*");
        milamina2.add(botonMultiplica);

        JButton boton0=new JButton("0");
        milamina2.add(boton0);
        JButton botonPunto=new JButton(".");
        milamina2.add(botonPunto);
        JButton botonIgual=new JButton("=");
        milamina2.add(botonIgual);
        JButton botonDivide=new JButton("/");
        milamina2.add(botonDivide);

        add(milamina2, BorderLayout.CENTER);
    }
}

```

Este será el resultado:



Los botones están ordenados por filas.

Ahora vamos a realizar un método más cómodo, para ello vamos a eliminar todo el código.

```

class LaminaCalculadora extends JPanel{
    public LaminaCalculadora() {
        setLayout(new BorderLayout());
        JButton pantalla =new JButton("0");
        pantalla.setEnabled(false);
        add(pantalla, BorderLayout.NORTH);

        //JPanel milamina2=new JPanel();
        milamina2=new JPanel();

        milamina2.setLayout(new GridLayout(4,4));
    }
}

```

```

        ponerBoton("7");
        ponerBoton("8");
        ponerBoton("8");
        ponerBoton("/");

        ponerBoton("4");
        ponerBoton("5");
        ponerBoton("6");
        ponerBoton("*");

        ponerBoton("1");
        ponerBoton("2");
        ponerBoton("3");
        ponerBoton("-");

        ponerBoton("0");
        ponerBoton(".");
        ponerBoton("=");
        ponerBoton("+");

        add(milamina2, BorderLayout.CENTER);
    }

    private void ponerBoton(String rotulo) {
        JButton boton=new JButton(rotulo);
        milamina2.add(boton);
    }

    private JPanel milamina2;
}

```

Hemos declarado una clase interna llamada ponerBoton.

La declaramos fuera de la clase intenta para que pueda tener acceso desde otras clases.

```

25 class LaminaCalculadora extends JPanel{
26     public LaminaCalculadora() {
27         setLayout(new BorderLayout());
28         JButton pantalla =new JButton("0");
29         pantalla.setEnabled(false);
30         add(pantalla, BorderLayout.NORTH);
31
32         //JPanel milamina2=new JPanel();
33         milamina2=new JPanel(); ←
34
35         milamina2.setLayout(new GridLayout(4,4));
36         ponerBoton("1");
37         ponerBoton("2");

```

El objeto milamina2 ya no hay que declararlo.

Este será el resultado:

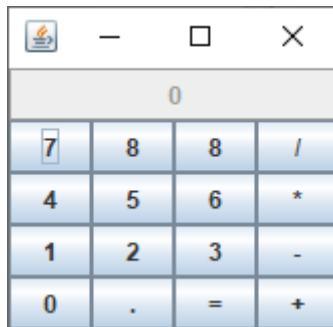


Layouts IV (VÍdeo 84)

```
16 class MarcoCalculadora extends JFrame{
17     public MarcoCalculadora() {
18         setTitle("Calculadora");
19         //setBounds(50,300,450,300);
20         LaminaCalculadora milamina=new LaminaCalculadora();
21         add(milamina);
22         pack(); ←
23     }
24 }
```

Pack(): El contenedor se tiene que adaptar al tamaño por defecto de lo que contiene.

Entonces no hace falta setBounds, este será el resultado:



Con los métodos getText y setText podemos capturar el texto que hay en el botón como establecer un nuevo texto en ese botón.

```
class LaminaCalculadora extends JPanel{
    public LaminaCalculadora() {
        setLayout(new BorderLayout());
        pantalla =new JButton("0");
        pantalla.setEnabled(false);
        add(pantalla, BorderLayout.NORTH);

        //JPanel milamina2=new JPanel();
        milamina2=new JPanel();

        ActionListener Insertar=new InsertaNumero();

        milamina2.setLayout(new GridLayout(4,4));
        ponerBoton("7",Insertar);
        ponerBoton("8",Insertar);
        ponerBoton("8",Insertar);
        //ponerBoton("/");

        ponerBoton("4",Insertar);
        ponerBoton("5",Insertar);
        ponerBoton("6",Insertar);
        //ponerBoton("*");

        ponerBoton("1",Insertar);
        ponerBoton("2",Insertar);
        ponerBoton("3",Insertar);
        //ponerBoton("-");

        ponerBoton("0",Insertar);
        //ponerBoton(".");
    }
}
```

Vamos a crear una instancia de InsertaNumero llamada Insertar.

Utilizaremos Insertar como segundo parámetro en los botones que tienen como título un número.


```

//ponerBoton("=");
//ponerBoton("+");

add(milamina2, BorderLayout.CENTER);

}

private void ponerBoton(String rotulo, ActionListener oyente) {
    JButton boton=new JButton(rotulo);
    boton.addActionListener(oyente); ← Ponemos a la escucha
    milamina2.add(boton); ← cada uno de los botones
}

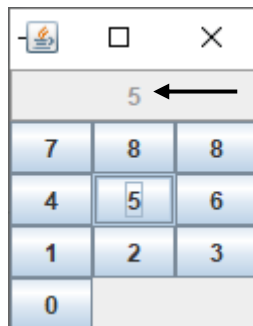
private class InsertaNumero implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String entrada=e.getActionCommand(); // Captura el texto
                                                // asociado al botón.
        pantalla.setText(entrada);
    }
    ← Al presionar un botón genera un evento y ejecuta el
    método actionPerformed, que pasa el valor a pantalla.

private JPanel milamina2;
private JButton pantalla; // Declaramos el objeto pantalla para
                          // que tenga acceso desde la clases interna
}
}

```

Si pulsamos el botón 5 este será el resultado:



Si pulsamos un segundo botón este se borra e imprime el segundo botón, esto tenemos que controlarlo.

```

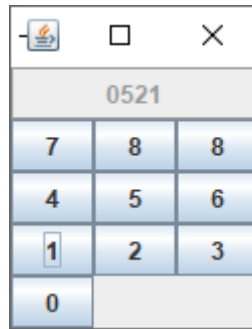
private class InsertaNumero implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String entrada=e.getActionCommand(); // Captura el texto asociado al botón.
        //pantalla.setText(entrada);
        pantalla.setText(pantalla.getText() + entrada); ←
    }
}
}

```

En la clase interna eliminamos la línea que hemos pasado a comentario y agregamos la siguiente línea.

Ahora cuando presionemos varios números, este será el resultado:



Observarás que el 0 del principio se mantiene.

```

70 private class InsertaNumero implements ActionListener{
71
72     @Override
73     public void actionPerformed(ActionEvent e) {
74         // TODO Auto-generated method stub
75         String entrada=e.getActionCommand(); // Captura el texto asociado al botón.
76         //pantalla.setText(entrada);
77         if(principio) {
78             pantalla.setText("");
79             principio=false;
80
81         }
82         pantalla.setText(pantalla.getText() + entrada);
83     }
84
85 }
86 private JPanel milamina2;
87 private JButton pantalla; // Declaramos el objeto pantalla para
88                             // que tenga acceso desde la clases interna
89 private boolean principio; ←
90 }

```

En la clase InsertarNumero definimos una variable de tipo boolean llamada principio que si no la declaramos por defecto es igual a false.

```

26 class LaminaCalculadora extends JPanel{
27     public LaminaCalculadora() {
28         principio=true; ←
29         setLayout(new BorderLayout());
30         pantalla =new JButton("0");
31         pantalla.setEnabled(false);
32         add(pantalla, BorderLayout.NORTH);
33     }
34 }

```

En la clase LaminaCalculadora a la variable principio de damos el valor true.

```

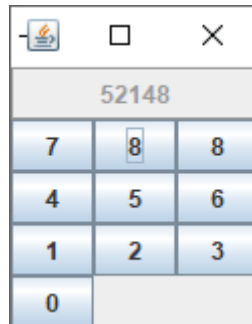
70 private class InsertaNumero implements ActionListener{
71
72     @Override
73     public void actionPerformed(ActionEvent e) {
74         // TODO Auto-generated method stub
75         String entrada=e.getActionCommand(); // Captura el texto asociado al botón.
76         //pantalla.setText(entrada);
77         if(principio) {
78             pantalla.setText("");
79             principio=false;
80
81         }
82         pantalla.setText(pantalla.getText() + entrada);
83     }
84 }

```

If(principio==true){
Es lo mismo.

En la clase InsertarNumero en el método actionPerformed con una condición comprobamos si es el primer número que introducimos para asignarle a pantalla.setText(""), y a la variable principio la pasamos de nuevo a false para que ya no pase por la condición.

Ese será el resultado:



Layouts V. (Vídeo 85)

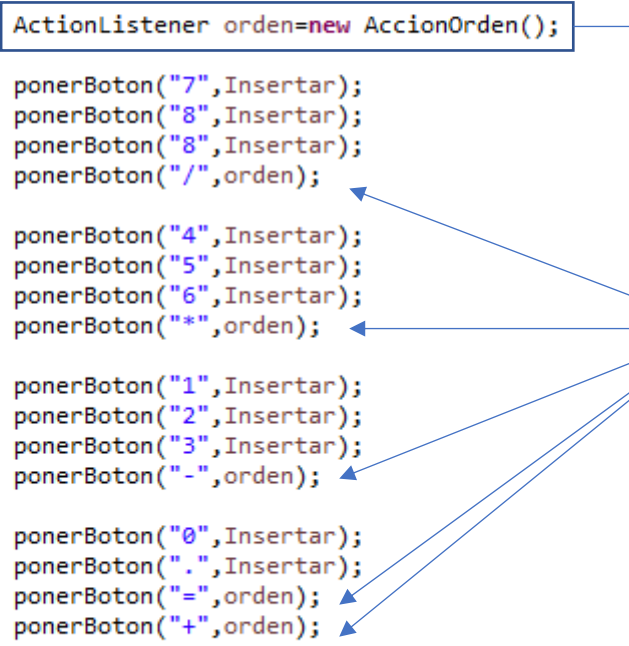
Ahora queremos que cuando le demos al botón de suma, resta, multiplicación o división cuando introduzcamos un número después del operador la pantalla tiene que empezar a escribir una numeración nueva.

Vamos a crear una clase interna en la clase llamada LaminaCalculadora llamada AccioOrden.

```
private class AccionOrden implements ActionListener{  
  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        // TODO Auto-generated method stub  
        principio = true;  
    }  
}
```

En el método correspondiente a la variable principio le damos el valor true.

```
26 class LaminaCalculadora extends JPanel{  
27     public LaminaCalculadora() {  
28         principio=true;  
29         setLayout(new BorderLayout());  
30         pantalla =new JButton("0");  
31         pantalla.setEnabled(false);  
32         add(pantalla, BorderLayout.NORTH);  
33  
34         //JPanel milamina2=new JPanel();  
35         milamina2=new JPanel();  
36  
37         ActionListener Insertar=new InsertaNumero();  
38  
39         milamina2.setLayout(new GridLayout(4,4));  
40  
41         ActionListener orden=new AccionOrden();  
42  
43         ponerBoton("7",Insertar);  
44         ponerBoton("8",Insertar);  
45         ponerBoton("8",Insertar);  
46         ponerBoton("/",orden);  
47  
48         ponerBoton("4",Insertar);  
49         ponerBoton("5",Insertar);  
50         ponerBoton("6",Insertar);  
51         ponerBoton("*",orden);  
52  
53         ponerBoton("1",Insertar);  
54         ponerBoton("2",Insertar);  
55         ponerBoton("3",Insertar);  
56         ponerBoton("-",orden);  
57  
58         ponerBoton("0",Insertar);  
59         ponerBoton(".",Insertar);  
60         ponerBoton("=",orden);  
61         ponerBoton("+",orden);  
62  
63         add(milamina2, BorderLayout.CENTER);  
64     }  
65 }  
~
```



Definimos un objeto de tipo ActionListener llamado orden de la clase AccionOrden()

Ahora si ejecutamos veremos que cada vez que presionemos un operador el siguiente valor numérico que se introduzca borrará de pantalla el valor introducido anteriormente.

Ahora queremos que la calculadora pueda realizar las operaciones de suma, resta, multiplicación y división.

Vamos a realizar los siguientes pasos:

```
100     private JPanel milamina2;
101     private JButton pantalla;
102
103     private boolean principio;
104     private double resultado; ←
105 }
```

Al final antes de la llave de cierre creamos una variable de tipo double llamada resultado, esta es double porque podrá almacenar números con decimales.

```
90 private class AccionOrden implements ActionListener{
91
92     @Override
93     public void actionPerformed(ActionEvent e) {
94         // TODO Auto-generated method stub
95         calcular(Double.parseDouble(pantalla.getText())); ←
96
97         principio = true;
98     }
}
```

En la clase AccionOrden vamos a llamar a la clase calcular donde le pasamos un parámetro en texto que tiene en pantalla convertido a doble.

Calcular esta marcada con una línea en rojo porque dicho método todavía no existe, lo tenemos que crear.

```
93     public void actionPerformed(ActionEvent e) {
94         // TODO Auto-generated method stub
95
96         String operacion=e.getActionCommand();
97         System.out.println(operacion);
98
99
100        calcular(Double.parseDouble(pantalla.getText()));
101
102        principio = true;
103    }
```

En la línea 96 le estamos diciendo que la variable operación almacene la operación que vamos a realizar +, -, *, / o =.

Para comprobar si es así en la línea 97 hacemos un System.out.println(operación);

Este será el resultado:

```
/
*
-
+
=
```

Ahora como vemos que funciona podemos borrar el contenido de la línea 97.

```

109     private JPanel milamina2;
110     private JButton pantalla;
111
112     private boolean principio;
113     private double resultado;
114     private String ultimaOperacion;
115 }

```

Antes de la última llave vamos a crear otra variable de tipo String llamada ultimaOperacion, con esta variable vamos a controlar si le hemos dado al botón igual ya que este tiene que mostrar el resultado final de la operación con la calculadora, en cambio con los operadores +, -, *, / tiene que esperar a introducir otro valor para borrar pantalla y poder escribir el siguiente valor.

```

93     public void actionPerformed(ActionEvent e) {
94         // TODO Auto-generated method stub
95
96         String operacion=e.getActionCommand();
97
98         calcular(Double.parseDouble(pantalla.getText()));
99         ultimaOperacion=operacion;
100
101         principio = true;
102     }

```

En el método actionPerformed a la variable ultimaOperación le asignamos el valor que tiene la variable operacion.

```

60     ponerBoton("=",orden);
61     ponerBoton("+",orden);
62
63     add(milamina2, BorderLayout.CENTER);
64
65     ultimaOperacion="=";
66
67 }

```

Al final de la clase LaminaCalculadora le decimos que ultimaOperacion es igual a "=", es decir inicializamos el valor de dicha variable.

```

106     public void calcular(double x) {
107         if(ultimaOperacion.equals("+")) {
108             resultado+=x;
109         }
110     }

```

En el método calcular le estamos diciendo que si pulsamos el botón de + el valor de x lo acumule a resultado.

```

94     @Override
95     public void actionPerformed(ActionEvent e) {
96         // TODO Auto-generated method stub
97
98         String operacion=e.getActionCommand();
99
100         calcular(Double.parseDouble(pantalla.getText()));
101         ultimaOperacion=operacion;
102
103         principio = true;
104     }

```

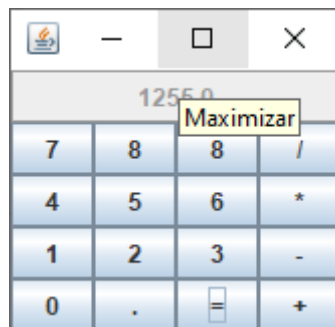
```

105
106 public void calcular(double x) {
107     if(ultimaOperacion.equals("+")) {
108         resultado+=x;
109     }
110     else if(ultimaOperacion.equals("-")) {
111         resultado-=x;
112     }
113     else if(ultimaOperacion.equals("*")) {
114         resultado*=x;
115     }
116     else if(ultimaOperacion.equals("/")) {
117         resultado/=x;
118     }
119     else if(ultimaOperacion.equals("=")) {
120         resultado=x;
121     }
122
123     pantalla.setText("" + resultado);
124 }

```

En el método actionPerformed controlamos el resto de operaciones y acumulando su valor en la variable resultado.

Ya puedes probar la calculadora:



Código completo de la calculadora.

```

package graficos;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Calculadora {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoCalculadora mimarco=new MarcoCalculadora();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mimarco.setVisible(true);
    }
}

class MarcoCalculadora extends JFrame{
    public MarcoCalculadora() {
        setTitle("Calculadora");
        //setBounds(50,300,450,300);
        LaminaCalculadora milamina=new LaminaCalculadora();

```

```

        add(milamina);
        pack();
    }
}
class LaminaCalculadora extends JPanel{
    public LaminaCalculadora() {
        principio=true;
        setLayout(new BorderLayout());
        pantalla =new JButton("0");
        pantalla.setEnabled(false);
        add(pantalla, BorderLayout.NORTH);

        //JPanel milamina2=new JPanel();
        milamina2=new JPanel();

        ActionListener Insertar=new InsertaNumero();

        milamina2.setLayout(new GridLayout(4,4));

        ActionListener orden=new AccionOrden();

        ponerBoton("7",Insertar);
        ponerBoton("8",Insertar);
        ponerBoton("8",Insertar);
        ponerBoton("/",orden);

        ponerBoton("4",Insertar);
        ponerBoton("5",Insertar);
        ponerBoton("6",Insertar);
        ponerBoton("*",orden);

        ponerBoton("1",Insertar);
        ponerBoton("2",Insertar);
        ponerBoton("3",Insertar);
        ponerBoton("-",orden);

        ponerBoton("0",Insertar);
        ponerBoton(".",Insertar);
        ponerBoton("=",orden);
        ponerBoton("+",orden);

        add(milamina2, BorderLayout.CENTER);

        ultimaOperacion="=";
    }

    private void ponerBoton(String rotulo, ActionListener oyente) {
        JButton boton=new JButton(rotulo);
        boton.addActionListener(oyente);
        milamina2.add(boton);
    }

    private class InsertaNumero implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub

```



```

String entrada=e.getActionCommand(); // Captura el texto
asociado al botón.
//pantalla.setText(entrada);
if(principio) {
    pantalla.setText("");
    principio=false;
}
pantalla.setText(pantalla.getText() + entrada);
}
}

private class AccionOrden implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub

        String operacion=e.getActionCommand();

        calcular(Double.parseDouble(pantalla.getText()));
        ultimaOperacion=operacion;

        principio = true;
    }

    public void calcular(double x) {
        if(ultimaOperacion.equals("+")) {
            resultado+=x;
        }
        else if(ultimaOperacion.equals("-")) {
            resultado-=x;
        }
        else if(ultimaOperacion.equals("*")) {
            resultado*=x;
        }
        else if(ultimaOperacion.equals("/")) {
            resultado/=x;
        }
        else if(ultimaOperacion.equals("=")) {
            resultado=x;
        }

        pantalla.setText("" + resultado);
    }
}

private JPanel milamina2;
private JButton pantalla; // Declaramos el objeto pantalla
// que tenga acceso
para desde la clases interna
private boolean principio;
private double resultado;
private String ultimaOperacion;
}

```

The image shows a course cover with a dark orange background. At the top left is a small Java logo. The main title 'CURSO JAVA' is in large white letters. To the right, the number '85' is in a yellow box. Below the title, 'COMPONENTES SWING' and 'Layouts V' are written in white. The background features a world map, the Java logo (a blue cup with orange steam), and the Eclipse logo (a grey sphere with horizontal lines). The word 'eclipse' is written in a light grey font at the bottom left, and 'Java' is written in a large orange font at the bottom right.

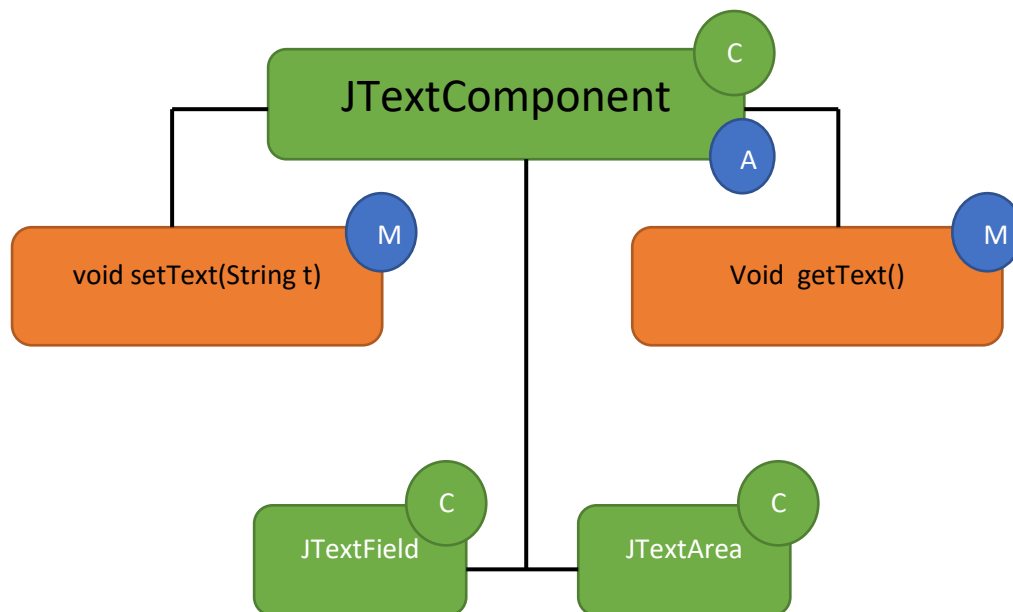
CURSO JAVA 85

COMPONENTES SWING
Layouts V

eclipse Java

Componentes Swing. Cuadros de texto I. (VÍdeo 86)

JTextField y JTextArea



```
1 package graficos;
2
3 import javax.swing.*;
4
5 public class PruebaTexto {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoTexto mimarco=new MarcoTexto();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        mimarco.setVisible(true);
12    }
13
14 }
15
16 class MarcoTexto extends JFrame{
17     public MarcoTexto() {
18         setBounds(600,300,600,350);
19         LaminaTexto milamina=new LaminaTexto();
20         add(milamina);
21         setVisible(true);
22     }
23 }
24
25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField();
28         add(campo1);
29     }
30 }
```

En la clase LaminaTexto en la línea 27 definimos un objeto de JTextField llamado campo1.

En la línea 28 la agregamos a LaminaTexto.

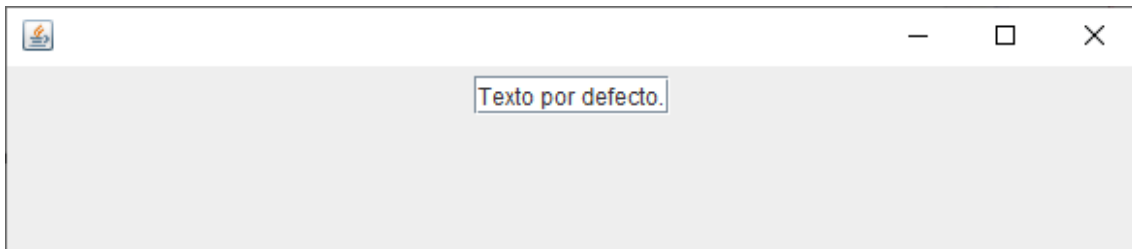
Este será el resultado:



Al no especificar ancho alguno este campo de texto es bastante inútil.

```
1 package graficos;
2
3 import javax.swing.*;
4
5 public class PruebaTexto {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoTexto mimarco=new MarcoTexto();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        mimarco.setVisible(true);
12    }
13
14 }
15
16 class MarcoTexto extends JFrame{
17     public MarcoTexto() {
18         setBounds(600,300,600,350);
19         LaminaTexto milamina=new LaminaTexto();
20         add(milamina);
21         setVisible(true);
22     }
23 }
24
25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField("Texto por defecto.");
28         add(campo1);
29     }
30 }
```

Si ponemos un texto este saldrá por defecto pudiéndolo modificar, este será el resultado:



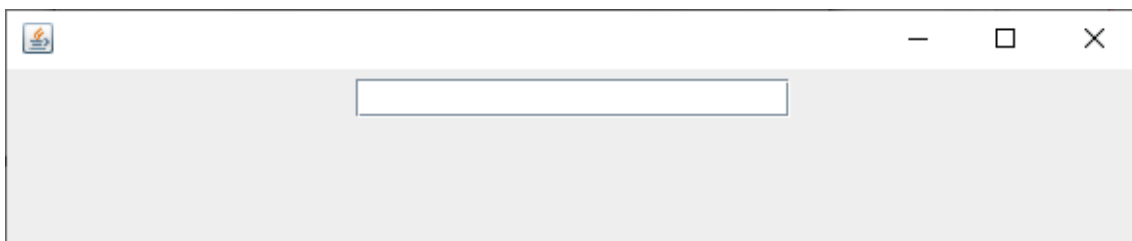
Con este constructor en ancho del campo se adapta a la longitud del texto.

```

1 package graficos;
2
3 import javax.swing.*;
4
5 public class PruebaTexto {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoTexto mimarco=new MarcoTexto();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        mimarco.setVisible(true);
12    }
13
14 }
15
16 class MarcoTexto extends JFrame{
17     public MarcoTexto() {
18         setBounds(600,300,600,350);
19         LaminaTexto milamina=new LaminaTexto();
20         add(milamina);
21         setVisible(true);
22     }
23 }
24
25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField(20);
28         add(campo1);
29     }
30 }

```

Este campo tendrá un ancho de 20 columnas, este será el resultado:

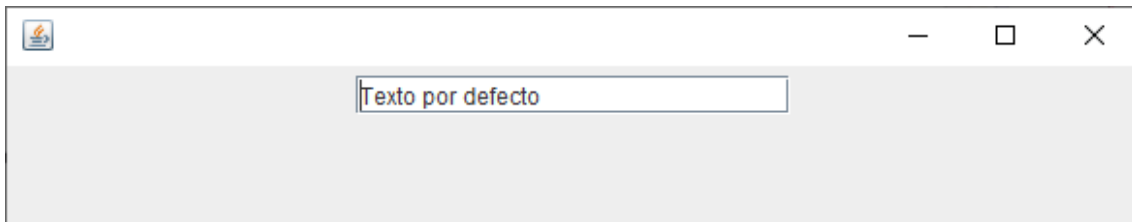


```

25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField("Texto por defecto",20);
28         add(campo1);
29     }
30 }

```

Con este constructor hay texto por defecto más una longitud de 20 columnas.



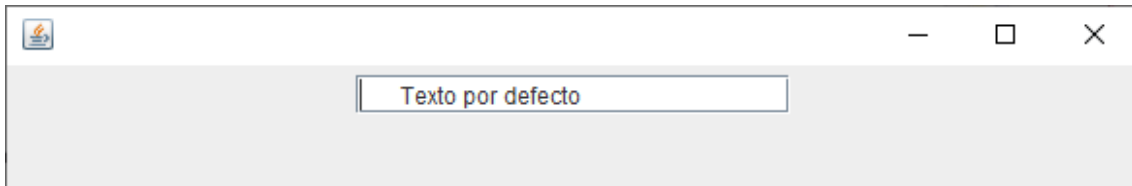
```
25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField("Texto por defecto",20)
28         add(campo1);
29         System.out.println(campo1.getText()); ←
30     }
31 }
```

Con `getText()` podemos recuperar el contenido de un campo de texto, cuando lo ejecutemos imprimirá el contenido del cuadro de texto por consola.

Texto por defecto

```
25 class LaminaTexto extends JPanel{
26     public LaminaTexto() {
27         JTextField campo1=new JTextField("  Texto por defecto",20);
28         add(campo1);
29         System.out.println(campo1.getText().trim()); ←
30     }
31 }
```

El método `trim()` omite los espacios en blanco.



Este será el resultado por consola.

Texto por defecto

Ahora vamos a agregar un botón en la venta y cuando lo pulsemos mande a consola el mensaje que tiene en la caja de texto.

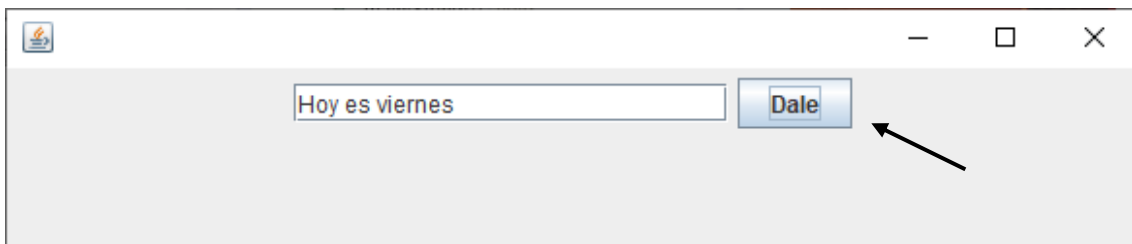
```
28 class LaminaTexto extends JPanel{
29     public LaminaTexto() {
30         campo1=new JTextField(20);
31         add(campo1);
32         JButton miboton=new JButton("Dale");
33         DameTexto mievento=new DameTexto();
34         miboton.addActionListener(mievento);
35         add(miboton);
36     }
37 }
38 }
```

```

39
40 private class DameTexto implements ActionListener{
41
42     @Override
43     public void actionPerformed(ActionEvent e) {
44         // TODO Auto-generated method stub
45         System.out.println(campo1.getText().trim());
46     }
47
48 }
49 private JTextField campo1;
50 }

```

Este será el resultado cuando ejecutemos.



Ponemos una frase seguido del botón dale, este será el resultado en consola.

```
Hoy es viernes
```

A continuación el código completo:

```

package graficos;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class PruebaTexto {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoTexto mimarco= new MarcoTexto();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mimarco.setVisible(true);
    }
}

class MarcoTexto extends JFrame{
    public MarcoTexto() {
        setBounds(600,300,600,350);
        LaminaTexto milamina=new LaminaTexto();
        add(milamina);
        setVisible(true);
    }
}

class LaminaTexto extends JPanel{
    public LaminaTexto() {
        campo1=new JTextField(20);

```

```
        add(campo1);
        JButton miboton=new JButton("Dale");
        DameTexto mievento=new DameTexto();
        miboton.addActionListener(mievento);
        add(miboton);
    }

    private class DameTexto implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            System.out.println(campo1.getText().trim());
        }
    }
    private JTextField campo1;
}
```



Componentes Swing. Cuadros de texto II. (Vídeo 87)

Ahora vamos a introducir un correo electrónico y compruebe si este es correcto o no.

```
package graficos;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class PruebaTexto {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoTexto mimarco= new MarcoTexto();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mimarco.setVisible(true);
    }
}

class MarcoTexto extends JFrame{
    public MarcoTexto() {
        setBounds(600,300,600,350);
        LaminaTexto milamina=new LaminaTexto();
        add(milamina);
        setVisible(true);
    }
}

class LaminaTexto extends JPanel{
    public LaminaTexto() {
        resultado=new JLabel();
        JLabel texto1 = new JLabel("Email: ");
        add(texto1);
        campo1=new JTextField(20);
        add(campo1);
        add(resultado);
        JButton miboton=new JButton("Comprobar");
        DameTexto mievento=new DameTexto();
        miboton.addActionListener(mievento);
        add(miboton);
    }
    private class DameTexto implements ActionListener{

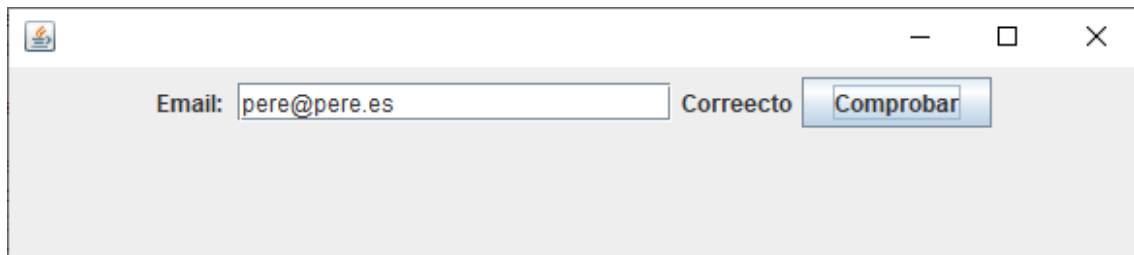
        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            int correcto=0;
            String email=campo1.getText().trim();
            for(int i=0;i<email.length();i++){
                if(email.charAt(i)=='@') {
                    correcto++;
                }
            }
            if(correcto !=1) {
```

```

        resultado.setText("Incorrecto");
    }
    else {
        resultado.setText("Correcto");
    }
}
}
private JTextField campo1;
private JLabel resultado;
}

```

Este será el resultado:



Cuando seleccionemos el botón comprobar que nos diga si el correo es correcto o incorrecto.

Hora queremos cambiar la ubicación el mensaje Correcto.

```

package graficos;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class PruebaTexto {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoTexto mimarco= new MarcoTexto();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mimarco.setVisible(true);
    }
}

class MarcoTexto extends JFrame{
    public MarcoTexto() {
        setBounds(600,300,600,350);
        LaminaTexto milamina=new LaminaTexto();
        add(milamina);
        setVisible(true);
    }
}

class LaminaTexto extends JPanel{
    public LaminaTexto() {
        setLayout(new BorderLayout());
        JPanel milamina2=new JPanel();
        milamina2.setLayout(new FlowLayout());
    }
}

```

```

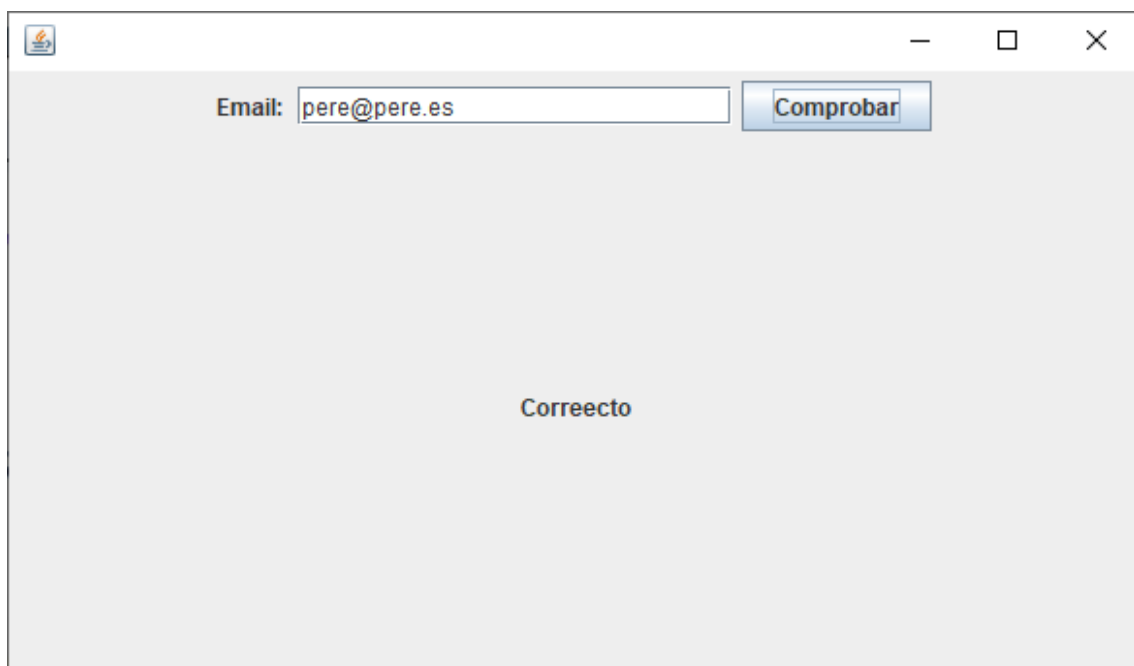
JLabel texto1 = new JLabel("Email: ");
milamina2.add(texto1);
resultado=new JLabel("",JLabel.CENTER);
campo1=new JTextField(20);
milamina2.add(campo1);
add(resultado, BorderLayout.CENTER);
JButton miboton=new JButton("Comprobar");
DameTexto mievento=new DameTexto();
miboton.addActionListener(mievento);
milamina2.add(miboton);
add(milamina2, BorderLayout.NORTH);
}
private class DameTexto implements ActionListener{

@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
int correcto=0;
String email=campo1.getText().trim();
for(int i=0;i<email.length();i++){
    if(email.charAt(i)=='@') {
        correcto++;
    }
}

if(correcto !=1) {
    resultado.setText("Incorrecto");
}
else {
    resultado.setText("Correcto");
}
}
}
private JTextField campo1;
private JLabel resultado;
}

```

Este será el resultado:





Ahora con lo aprendido te planteo el siguiente proyecto, introduciendo el número del dni que te calcule la letra.

La formula para calcular la letra del DNI y obtener el Nif es la siguiente:

Tomamos el número completo de hasta 8 cifras de nuestro DNI, lo dividimos entre 23 y nos quedamos con el resto de dicha división, o dicho de otro modo, calculamos el módulo 23 del DNI.

El resultado anterior es un número entre 0 y 22. A cada uno de estos posibles números le corresponde una letra, según la siguiente tabla:

RESTO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

En la siguiente página tienes el correspondiente código:

Puedes utilizar el código del proyecto anterior y realizar las modificaciones oportunas.

```

class MarcoNif extends JFrame{
    public MarcoNif() {

        setTitle("Para calcular letra NIF");
        setBounds(600,300,600,350);
        LaminaNif milamina=new LaminaNif();
        add(milamina);
        setVisible(true);
    }
}

class LaminaNif extends JPanel{
    public LaminaNif() {
        setLayout(new BorderLayout());
        JPanel milamina2=new JPanel();
        milamina2.setLayout(new FlowLayout());
        JLabel texto1=new JLabel("Introduce Nif: ");
        milamina2.add(texto1);
        resultado=new JLabel("", JLabel.CENTER);
        campo1=new JTextField(20);
        JButton miboton=new JButton("Calcular letra");
        milamina2.add(campo1);
        milamina2.add(miboton);
        add(resultado, BorderLayout.CENTER);
        DameNif minif=new DameNif();
        miboton.addActionListener(minif);
        add(milamina2,BorderLayout.NORTH);
    }

    private class DameNif implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            valor=campo1.getText();
            valor1=Integer.parseInt(valor);

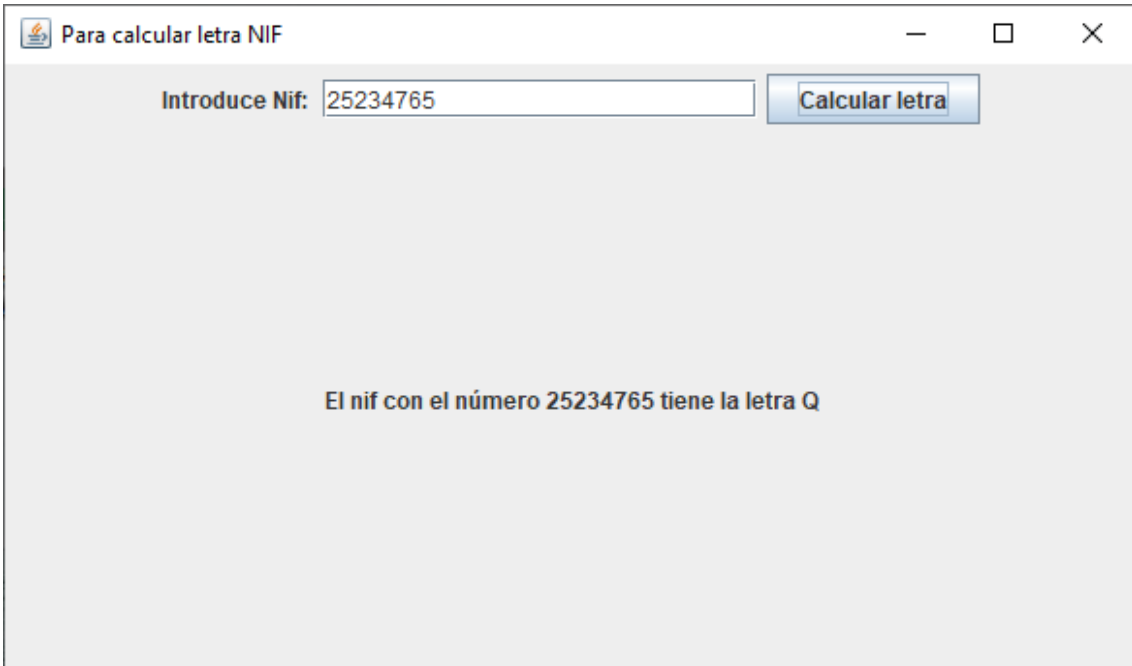
            resultado.setText("El nif con el número " + valor +
" tiene la letra " +
Character.toString(letras[valor1%23]));
        }

    }

    private String valor;
    private int valor1;
    private JLabel resultado;
    private JTextField campo1;
    char letras[] =
{'T','R','W','A','G','M','Y','F','P','D','X','B','N','J','Z','S','Q','V','H',
'L','C','K','E'};
}

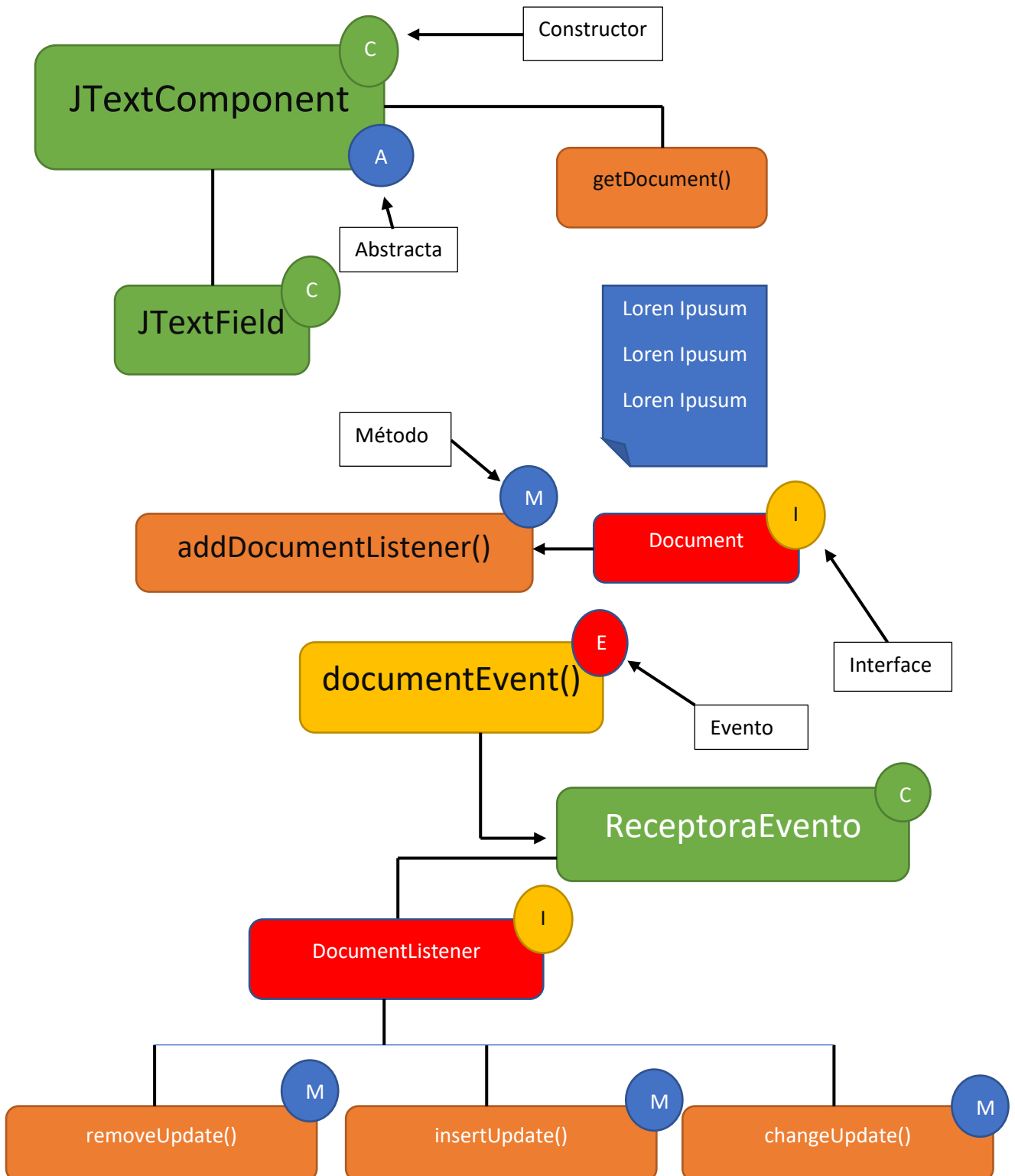
```

Este será el resultado cuando ejecutemos el programa.



Componentes Swing. Eventos de cuadros de texto. (Vídeo 88)

Cambios en un JTextField ¿Cómo gestionamos el evento?



Cuando se aplica el método `getDocument()` en un `JTextField` consigues que te devuelva un `document`, que es un modelo o una representación del texto que hay dentro del `JTextField`.

Cuando tienes un cuadro de texto y has escrito algo si aplicas el `getDocument()` a ese cuadro de texto consigues crear un objeto de tipo `document` en el cual contiene una representación de ese texto.

Document es una interface.

```
package graficos;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
public class PruebaDocument {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoPrueba mimarco=new MarcoPrueba();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}

class MarcoPrueba extends JFrame{
    public MarcoPrueba() {
        setBounds(500,300,500,350);
        LaminaPrueba milamina=new LaminaPrueba();
        add(milamina);
        setVisible(true);
    }
}

class LaminaPrueba extends JPanel{
    public LaminaPrueba() {
        JTextField micampo=new JTextField(20);
        EscuchaTexto el_evento= new EscuchaTexto();
        Document midoc=micampo.getDocument();
        midoc.addDocumentListener(el_evento);
        add(micampo);
    }

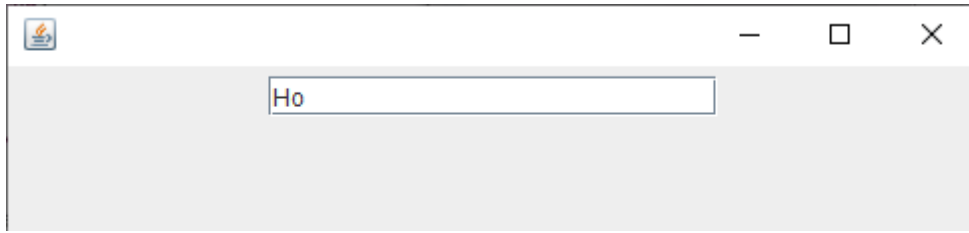
    private class EscuchaTexto implements DocumentListener{

        @Override
        public void insertUpdate(DocumentEvent e) {
            // TODO Auto-generated method stub
            System.out.println("Has insertado texto.");
        }

        @Override
        public void removeUpdate(DocumentEvent e) {
            // TODO Auto-generated method stub
            System.out.println("Has borrado texto.");
        }

        @Override
        public void changedUpdate(DocumentEvent e) {
            // TODO Auto-generated method stub
        }
    }
}
}
```

Este será el resultado:



Escribe la palabra Hola y luego borra la 'a' y la 'l', este será el resultado en consola.

```
Has insertado texto.  
Has insertado texto.  
Has insertado texto.  
Has insertado texto.  
Has borrado texto.  
Has borrado texto.
```

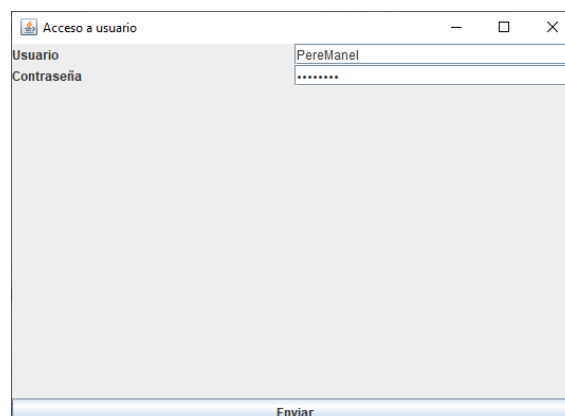


Componentes Swing. Eventos de cuadros de texto II. (Vídeo 89)

En este capítulo vamos a crear una aplicación que nos pida nombre de usuario y una contraseña que tiene que tener una longitud mínima de 8 y máxima de 12.

```
1 package graficos;
2 import java.awt.*;
3
4 public class CampoPassword {
5
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoPassword mimarco=new MarcoPassword();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11    }
12
13 }
14
15 class MarcoPassword extends JFrame{
16     public MarcoPassword(){
17         setTitle("Acceso a usuario");
18         setBounds(400,300,550,400);
19         LaminaPassword milamina=new LaminaPassword();
20         add(milamina);
21         setVisible(true);
22     }
23
24 }
25 class LaminaPassword extends JPanel{
26     public LaminaPassword() {
27         setLayout(new BorderLayout());
28         JPanel lamina_superior=new JPanel();
29         lamina_superior.setLayout(new GridLayout(2,2));
30         add(lamina_superior, BorderLayout.NORTH);
31         JLabel etiqueta1=new JLabel("Usuario");
32         JLabel etiqueta2=new JLabel("Contraseña");
33         JTextField c_usuario=new JTextField(15);
34         JPasswordField c_contra=new JPasswordField(15);
35         lamina_superior.add(etiqueta1);
36         lamina_superior.add(c_usuario);
37         lamina_superior.add(etiqueta2);
38         lamina_superior.add(c_contra);
39         JButton enviar=new JButton("Enviar");
40         add(enviar, BorderLayout.SOUTH);
41     }
42 }
```

Este será el resultado de la parte gráfica:



Ahora vamos a crear la clase receptora de los eventos.

```
27 public LaminaPassword() {
28     setLayout(new BorderLayout());
29     JPanel lamina_superior=new JPanel();
30     lamina_superior.setLayout(new GridLayout(2,2));
31     add(lamina_superior, BorderLayout.NORTH);
32     JLabel etiqueta1=new JLabel("Usuario");
33     JLabel etiqueta2=new JLabel("Contraseña");
34     JTextField c_usuario=new JTextField(15);
35
36     Comprueba_pass mievento=new Comprueba_pass();
37
38     c_contra=new JPasswordField(15);
39
40     c_contra.getDocument().addDocumentListener(mievento);
41
42     lamina_superior.add(etiqueta1);
43     lamina_superior.add(c_usuario);
44     lamina_superior.add(etiqueta2);
45     lamina_superior.add(c_contra);
46     JButton enviar=new JButton("Enviar");
47     add(enviar, BorderLayout.SOUTH);
48 }
49 private class Comprueba_pass implements DocumentListener{
50
51     @Override
52     public void insertUpdate(DocumentEvent e) {
53         // TODO Auto-generated method stub
54         char[] contrasena;
55         contrasena = c_contra.getPassword();
56         if(contrasena.length<8 || contrasena.length>12){
57             c_contra.setBackground(Color.RED);
58         }
59         else {
60             c_contra.setBackground(Color.WHITE);
61         }
62     }
63
64     @Override
65     public void removeUpdate(DocumentEvent e) {
66         // TODO Auto-generated method stub
67
68     }
69
70     @Override
71     public void changedUpdate(DocumentEvent e) {
72         // TODO Auto-generated method stub
73
74     }
75
76 }
77 JPasswordField c_contra;
78 }
```

Creamos la clase interna llamada Comprueba_pass en la clase LaminaPassword.

Definimos la variable c_contra dentro de la clase LaminaPassword.

```

26 class LaminaPassword extends JPanel{
27     public LaminaPassword() {
28         setLayout(new BorderLayout());
29         JPanel lamina_superior=new JPanel();
30         lamina_superior.setLayout(new GridLayout(2,2));
31         add(lamina_superior, BorderLayout.NORTH);
32         JLabel etiqueta1=new JLabel("Usuario");
33         JLabel etiqueta2=new JLabel("Contraseña");
34         JTextField c_usuario=new JTextField(15);
35
36         Comprueba_pass mivento=new Comprueba_pass();
37
38         c_contra=new JPasswordField(15);
39
40         c_contra.getDocument().addDocumentListener(mivento);
41
42         lamina_superior.add(etiqueta1);
43         lamina_superior.add(c_usuario);
44         lamina_superior.add(etiqueta2);
45         lamina_superior.add(c_contra);
46         JButton enviar=new JButton("Enviar");
47         add(enviar, BorderLayout.SOUTH);
48     }

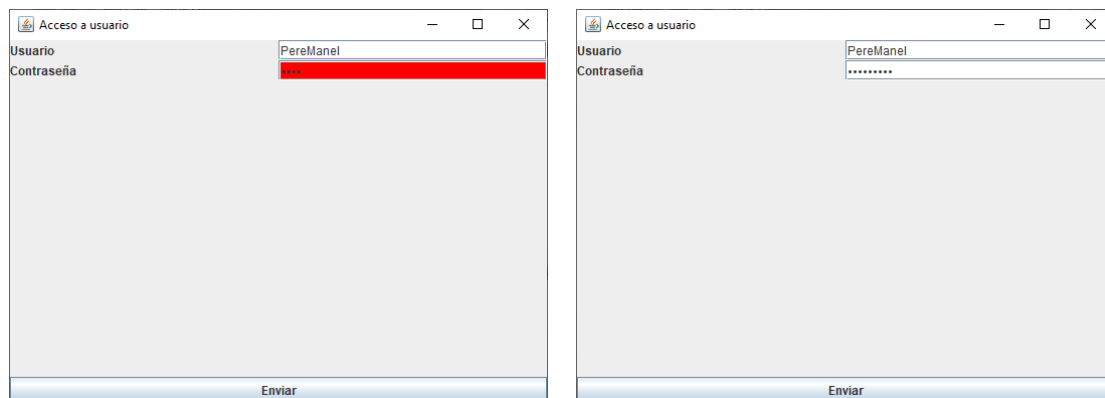
```

En la línea 38 quitamos JPasswordField ya que la hemos definido al final de la clase LaminaPassword.

En la línea 36 definimos un objeto de tipo Comprueba_pass llamado mivento.

En la línea 40 ponemos la variable c_contra a la escucha.

Este será el resultado:



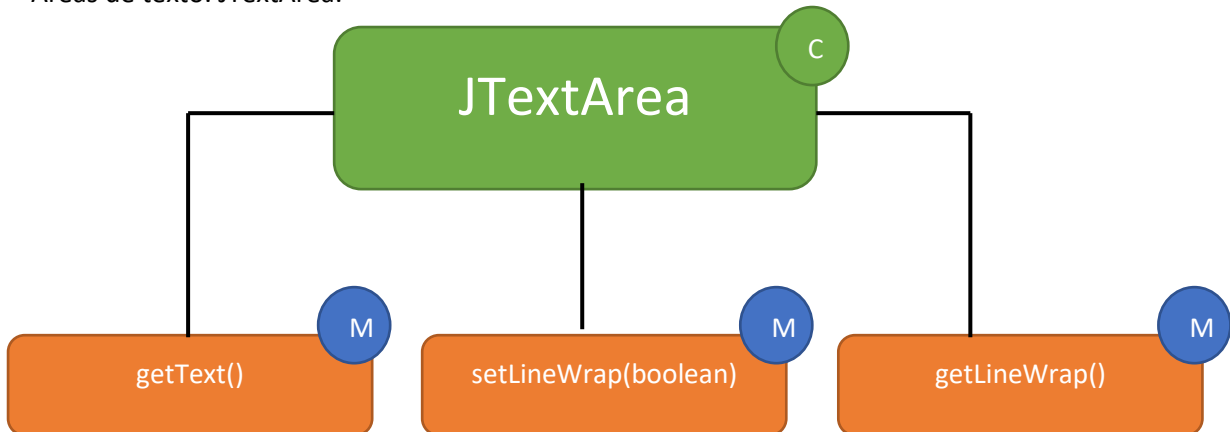
Si la contraseña tiene menos de 8 caracteres o más de 12 mostrará un color rojo y está entre los valores correctos de 8 a 12 la caja de texto se mostrará blanca.

A slide with a dark orange background featuring a world map. The slide contains the following text and logos:

- Top left: A small orange logo with the word "Java" written vertically.
- Top center: The text "CURSO JAVA" in large, bold, white capital letters.
- Top right: The number "89" in white, enclosed in a yellow square.
- Center: The text "COMPONENTES SWING" in bold white capital letters, with "Cambios en un campo de texto II" in smaller white text below it.
- Bottom left: The word "eclipse" in a light, lowercase font.
- Bottom right: The Java logo (a blue coffee cup with orange steam) and the word "Java" in its signature orange font.

Componentes Swing. Área de texto I. (Vídeo 90)

Áreas de texto. JTextArea.



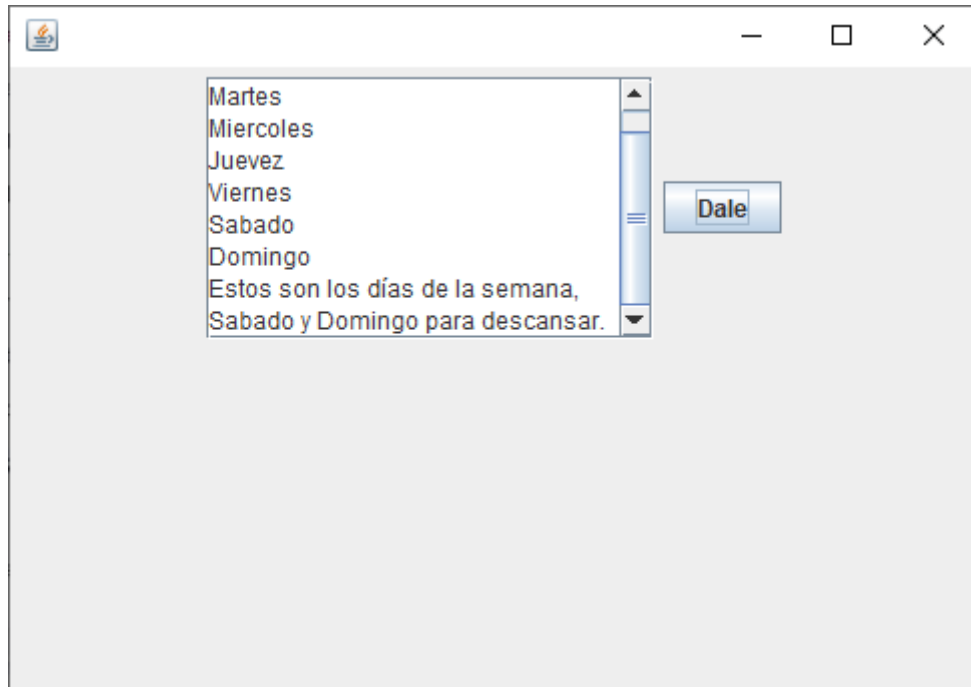
```
1 package graficos;
2
3 import java.awt.event.*;
4
5 import javax.swing.*;
6
7 public class EjemploArea {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         MarcoArea mimarco=new MarcoArea();
12         mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13     }
14 }
15
16 class MarcoArea extends JFrame{
17     public MarcoArea() {
18         setBounds(500,300,500,350);
19         LaminaArea milamina=new LaminaArea();
20         add(milamina);
21         setVisible(true);
22     }
23 }
24 class LaminaArea extends JPanel{
25     public LaminaArea() {
26         miarea=new JTextArea(8,20);
27         JScrollPane laminaBarra=new JScrollPane(miarea);
28         miarea.setLineWrap(true);
29         add(laminaBarra);
30         JButton miboton=new JButton("Dale");
31         miboton.addActionListener(new GestionArea()); ←
32         add(miboton);
33     }
34     private class GestionArea implements ActionListener{
35         @Override
36         public void actionPerformed(ActionEvent e) {
37             // TODO Auto-generated method stub
38             System.out.println(miarea.getText());
39         }
40     }
41     private JTextArea miarea;
42 }
```

En la línea 27 creamos una nueva lamina JScrollPane donde agregaremos el texto con área y este al escribir cuando pase de las filas se genera la barra de desplazamiento automáticamente.

En la línea 28 le decimos que aparezca la barra de desplazamiento cuando sea necesario, si fuere false la caja de texto iría creciendo.

En la clase interna GestionaArea imprime por consola cuando seleccionamos el botón, ya que en la línea 31 está permaneciendo a la escucha.

Cuando ejecutamos este será el resultado:



Una vez escrito el texto presionado el botón "Dale" este será el resultado por consola:

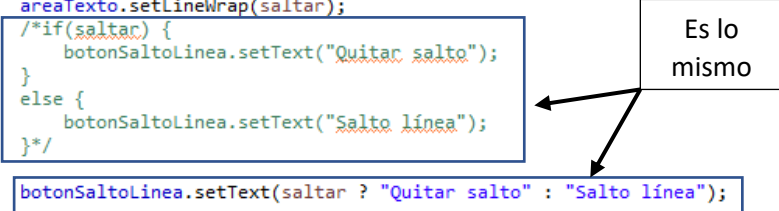
```
Lunes
Martes
Miercoles
Juevez
Viernes
Sabado
Domingo
Estos son los días de la semana,
Sabado y Domingo para descansar.
```





Componentes Swing. Áreas de texto II. (Vídeo 91)

```
1 package graficos;
2
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 import javax.swing.*;
7
8 public class PruebaArea {
9
10 public static void main(String[] args) {
11     // TODO Auto-generated method stub
12     MarcoPruebaArea mimarco = new MarcoPruebaArea();
13     mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14     mimarco.setVisible(true);
15 }
16 }
17 class MarcoPruebaArea extends JFrame{
18 public MarcoPruebaArea() {
19     setTitle("Probando Área de texto");
20     setBounds(500,300,500,350);
21     setLayout(new BorderLayout());
22     laminaBotones=new JPanel();
23     botonInsertar=new JButton("Insertar");
24     botonInsertar.addActionListener(new ActionListener() {
25
26         @Override
27         public void actionPerformed(ActionEvent e) {
28             // TODO Auto-generated method stub
29             areaTexto.append("En un lugar de la Mancha en cuyo nombre no quiero acordarme...");
30         }
31     });
32     laminaBotones.add(botonInsertar);
33     botonSaltoLinea=new JButton("Salto Línea");
34     botonSaltoLinea.addActionListener(new ActionListener(){
35
36         @Override
37         public void actionPerformed(ActionEvent e) {
38             // TODO Auto-generated method stub
39             boolean saltar=!areaTexto.getLineWrap();
40             areaTexto.setLineWrap(saltar);
41             /*if(saltar) {
42                 botonSaltoLinea.setText("Quitar salto");
43             }
44             else {
45                 botonSaltoLinea.setText("Salto línea");
46             }*/
47             botonSaltoLinea.setText(saltar ? "Quitar salto" : "Salto línea");
48         }
49     });
50 }
51
52     laminaBotones.add(botonSaltoLinea);
53     add(laminaBotones, BorderLayout.SOUTH);
54     areaTexto=new JTextArea(8,20);
55     laminaConBarras=new JScrollPane(areaTexto);
56     add(laminaConBarras, BorderLayout.CENTER);
57 }
58 private JPanel laminaBotones;
59 private JButton botonInsertar, botonSaltoLinea;
60 private JTextArea areaTexto;
61 private JScrollPane laminaConBarras;
62 }
```



Este proyecto se ha realizado todo en una misma clase, no es aconsejable pero es otra forma de programar.

Vamos a realizar el mismo código con el método tradicional.

Intenta ayudarte con ejemplos anteriores, con este ejercicio te ayudará a comprender más la programación orientada a objetos, la solución está en la página siguiente.

```

package pere_manel;

import java.awt.BorderLayout;
import java.awt.event.*;
import javax.swing.*;

public class PruebaArea {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoPruebaArea mimarco = new MarcoPruebaArea();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MarcoPruebaArea extends JFrame{
    public MarcoPruebaArea() {
        setTitle("Probando Área de texto");
        setBounds(500,300,500,350);
        LaminaPruebaArea milamina=new LaminaPruebaArea();
        add(milamina);
        setVisible(true);
    }
    private JScrollPane laminaConBarras;
}

class LaminaPruebaArea extends JPanel{
    public LaminaPruebaArea() {
        setLayout(new BorderLayout());
        laminaBotones=new JPanel();
        botonInsertar=new JButton("Insertar");
        botonInsertar.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                areaTexto.append("en un lugar de la Mancha, en cuyo
nombre no quiero acordarme...");
            }
        });
        laminaBotones.add(botonInsertar);
        botonSaltoLinea=new JButton("Salto línea");
        botonSaltoLinea.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                boolean saltar=!areaTexto.getLineWrap();
                areaTexto.setLineWrap(saltar);
                botonSaltoLinea.setText(saltar ? "Quitar salto" :
"Salto línea");
            }
        });
        laminaBotones.add(botonSaltoLinea);
        add(laminaBotones, BorderLayout.SOUTH);

        areaTexto=new JTextArea(8,20);
        JScrollPane laminaConBarras=new JScrollPane(areaTexto);

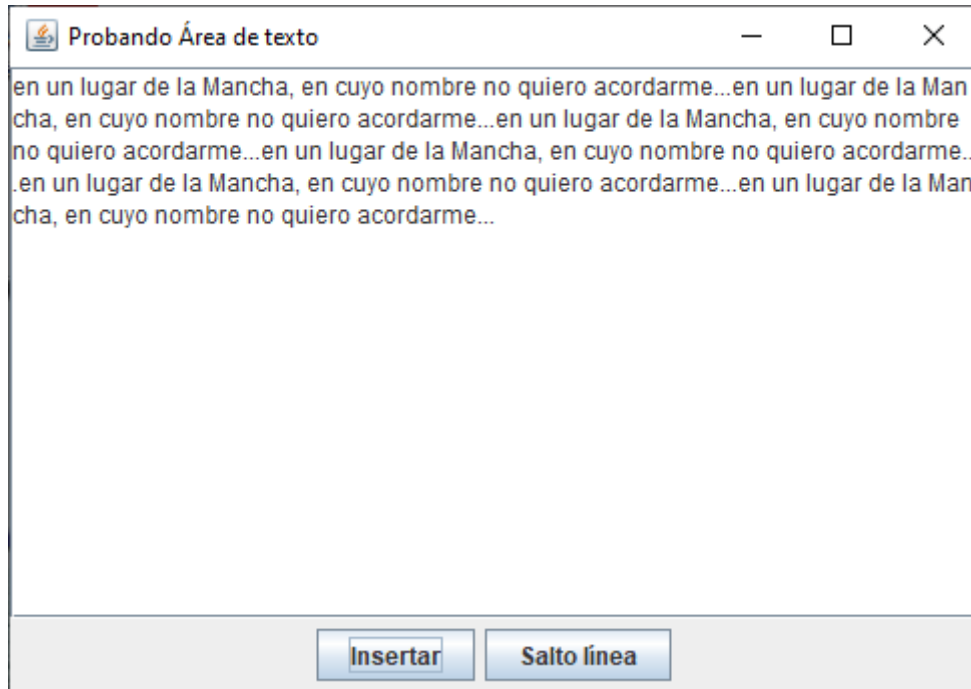
```

```

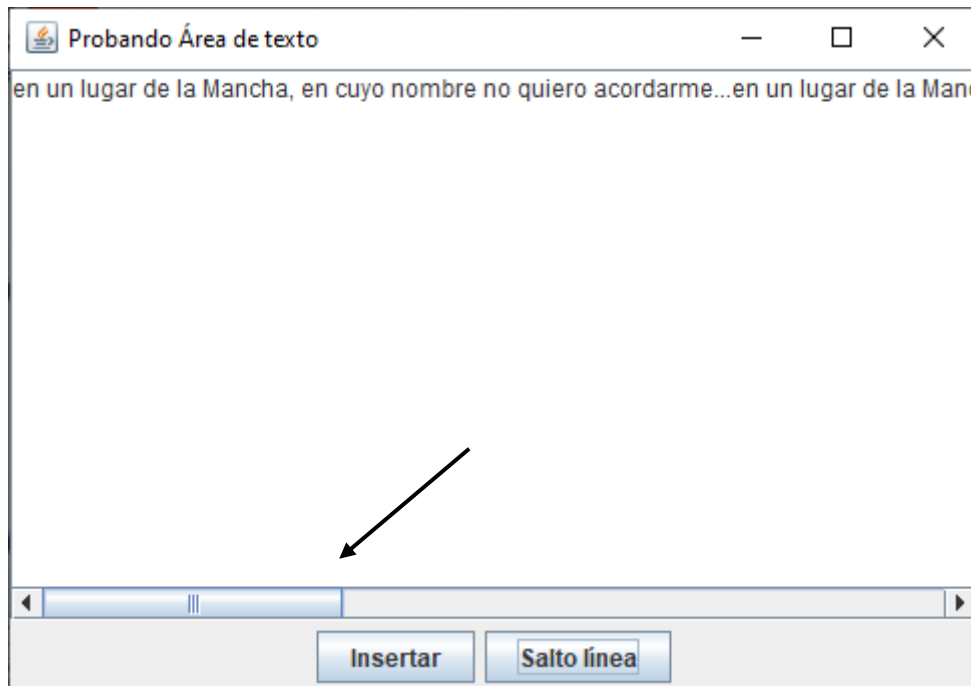
        areaTexto.setLineWrap(true);
        laminaConBarras=new JScrollPane(areaTexto);
        add(laminaConBarras, BorderLayout.CENTER);
    }
    private JPanel laminaBotones;
    private JButton botonInsertar, botonSaltoLinea;
    private JTextArea areaTexto;
}

```

Cuando ejecutemos este será el resultado:



Seleccionaremos el botón insertar cuatro veces, a continuación seleccionaremos el botón Salto línea.

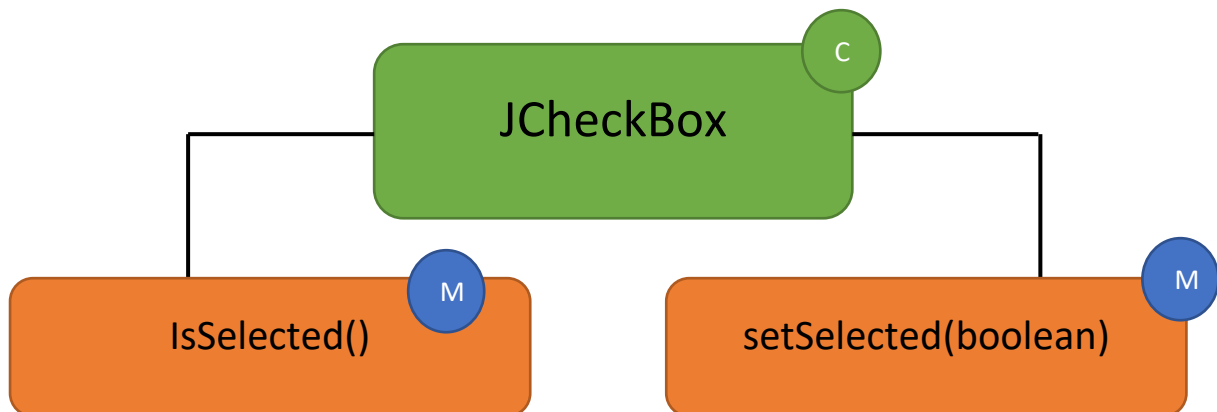


Aparecerá una barra de desplazamiento horizontal para desplazarnos por toda la línea.



Componentes Swing CheckBox. (VÍdeo 92)

Casillas de Verificación JCheckBox.



En este capítulo vamos a realizar una ventana con una caja de texto y dos casillas de verificación, al activar una el texto pasa en negrita y la otra en cursiva.

Vamos a escribir el siguiente código:

```
package graficos;

import java.awt.*;
import java.awt.Font;
import java.awt.event.*;

import javax.swing.*;

public class PruebaChecks {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoCheck mimarco=new MarcoCheck();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MarcoCheck extends JFrame{
    public MarcoCheck() {
        setBounds(550,300,550,350);
        setVisible(true);
        LaminaCheck milamina=new LaminaCheck();
        add(milamina);
    }
}

class LaminaCheck extends JPanel{
    public LaminaCheck() {
        setLayout(new BorderLayout());
        Font miletra=new Font("Serif", Font.PLAIN, 24);
        texto=new JLabel("En un lugar de la mancha de cuyo nombre...");
        texto.setFont(miletra);
        JPanel laminatexto=new JPanel();
        laminatexto.add(texto);
        add(laminatexto, BorderLayout.CENTER);
        check1=new JCheckBox("Negrita");
        check2=new JCheckBox("Cursiva");
```

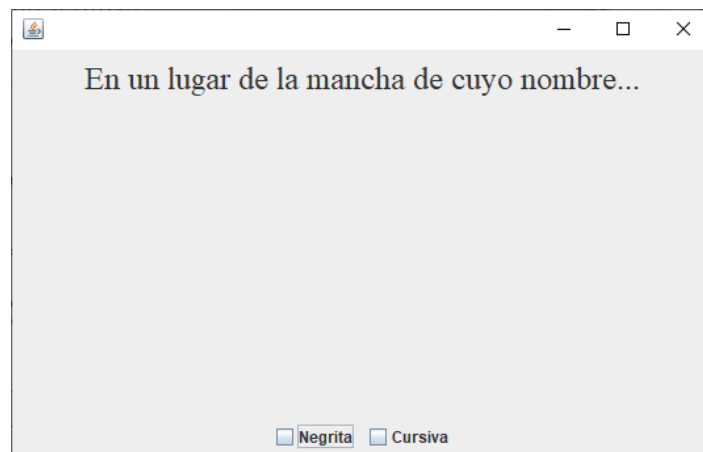
```

        check1.addActionListener(new ManejaChecks());
        check2.addActionListener(new ManejaChecks());
        JPanel laminachecks=new JPanel();
        laminachecks.add(check1);
        laminachecks.add(check2);
        add(laminachecks, BorderLayout.SOUTH);
    }
    private class ManejaChecks implements ActionListener{

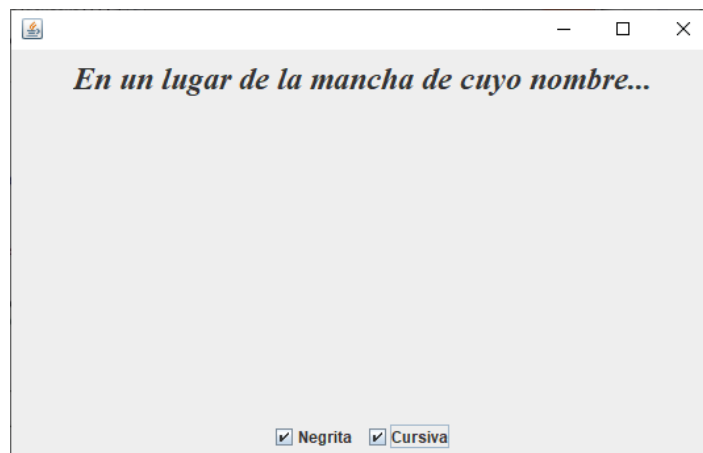
        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            int tipo=0;
            if(check1.isSelected()) tipo+=Font.BOLD; //BOLD = 1
            if(check2.isSelected()) tipo+=Font.ITALIC; // ITALIC = 2
            texto.setFont(new Font("Serif", tipo, 24));
        }
    }
    private JLabel texto;
    private JCheckBox check1, check2;
}

```

Si lo ejecutamos este será el resultado:



Ahora activa la Negrita, observa el texto y por último activa la cursiva.



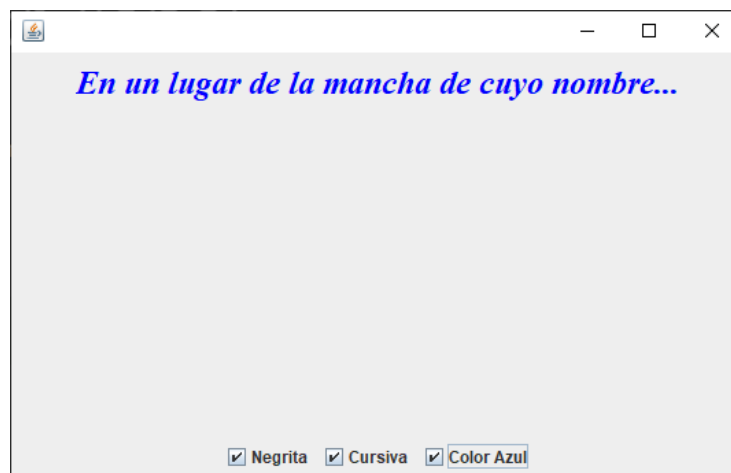
Ahora agrega un nuevo check que diga "Color azul" y cuando lo selecciones cambie a color azul y si lo desactivas color negro, en la próxima pagina tienes las modificaciones del código:

```

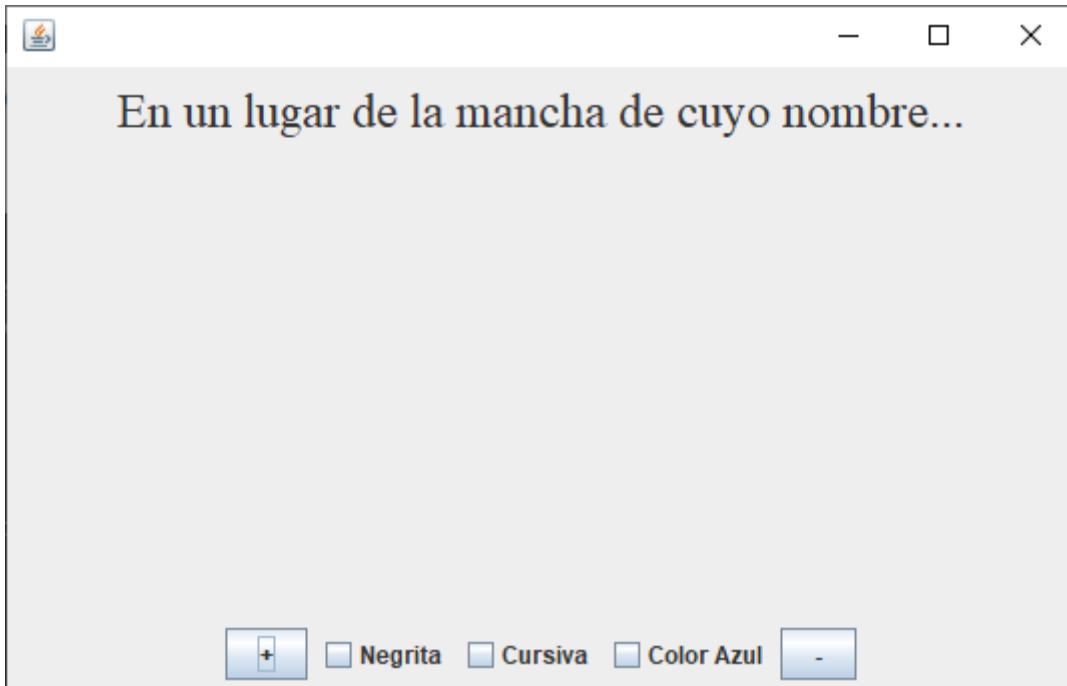
25 class LaminaCheck extends JPanel{
26     public LaminaCheck() {
27         setLayout(new BorderLayout());
28         Font miletra=new Font("Serif", Font.PLAIN, 24);
29         texto=new JLabel("En un lugar de la mancha de cuyo nombre...");
30         texto.setFont(miletra);
31         JPanel laminatexto=new JPanel();
32         laminatexto.add(texto);
33         add(laminatexto, BorderLayout.CENTER);
34         check1=new JCheckBox("Negrita");
35         check2=new JCheckBox("Cursiva");
36         check3=new JCheckBox("Color Azul"); ←
37         check1.addActionListener(new ManejaChecks());
38         check2.addActionListener(new ManejaChecks());
39         check3.addActionListener(new ManejaChecks()); ←
40         JPanel laminachecks=new JPanel();
41         laminachecks.add(check1);
42         laminachecks.add(check2);
43         laminachecks.add(check3); ←
44         add(laminachecks, BorderLayout.SOUTH);
45     }
46     private class ManejaChecks implements ActionListener{
47
48         @Override
49         public void actionPerformed(ActionEvent e) {
50             // TODO Auto-generated method stub
51             int tipo=0;
52             if(check1.isSelected()) tipo+=Font.BOLD; //BOLD = 1
53             if(check2.isSelected()) tipo+=Font.ITALIC; // ITALIC = 2
54             texto.setFont(new Font("Serif", tipo, 24));
55             if(check3.isSelected()) {
56                 texto.setForeground(Color.BLUE);
57             }
58             else {
59                 texto.setForeground(Color.BLACK);
60             }
61         }
62     }
63 }
64 private JLabel texto;
65 private JCheckBox check1, check2, check3;
66 }

```

Este será el resultado:



Ahora vamos a modificar el proyecto añadiendo un botón + y otro -.



El más aumenta el tamaño de la fuente en 1 y el menos disminuye la fuente en 1.

Este es el código:

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    MarcoCheck mimarco=new MarcoCheck();
    mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

class MarcoCheck extends JFrame{
    public MarcoCheck() {
        setBounds(550,300,550,350);
        setVisible(true);
        LaminaCheck milamina=new LaminaCheck();
        add(milamina);
    }
}

class LaminaCheck extends JPanel{
    public LaminaCheck() {
        setLayout(new BorderLayout());
        Font miletra=new Font("Serif", Font.PLAIN, 24);
        texto=new JLabel("En un lugar de la mancha de cuyo nombre...");
        texto.setFont(miletra);
        JPanel laminatexto=new JPanel();
        laminatexto.add(texto);
        add(laminatexto, BorderLayout.CENTER);
        botonmas=new JButton("+"); ←
        check1=new JCheckBox("Negrita");
        check2=new JCheckBox("Cursiva");
        check3=new JCheckBox("Color Azul");
        botonmenos=new JButton("-"); ←
    }
}
```



```

botonmas.addActionListener(new ManejaChecks()); ←
check1.addActionListener(new ManejaChecks());
check2.addActionListener(new ManejaChecks());
check3.addActionListener(new ManejaChecks());
botonmenos.addActionListener(new ManejaChecks()); ←

JPanel laminachecks=new JPanel();
laminachecks.add(botonmas); ←
laminachecks.add(check1);
laminachecks.add(check2);
laminachecks.add(check3);
laminachecks.add(botonmenos); ←
add(laminachecks, BorderLayout.SOUTH);
}
private class ManejaChecks implements ActionListener{

@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
int tipo=0;
if(e.getSource()==botonmas) {
    tamano++;
}
if(e.getSource()==botonmenos) {
    tamano--;
}
if(check1.isSelected()) tipo+=Font.BOLD; //BOLD = 1
if(check2.isSelected()) tipo+=Font.ITALIC; // ITALIC = 2
texto.setFont(new Font("Serif", tipo, tamano));
if(check3.isSelected()) {
    texto.setForeground(Color.BLUE); ←
}
else {
    texto.setForeground(Color.BLACK);
}
}
}
private JLabel texto;
private JCheckBox check1, check2, check3, checkmas, checkmenos;
private JButton botonmas, botonmenos;
int tamano=24; ←
}

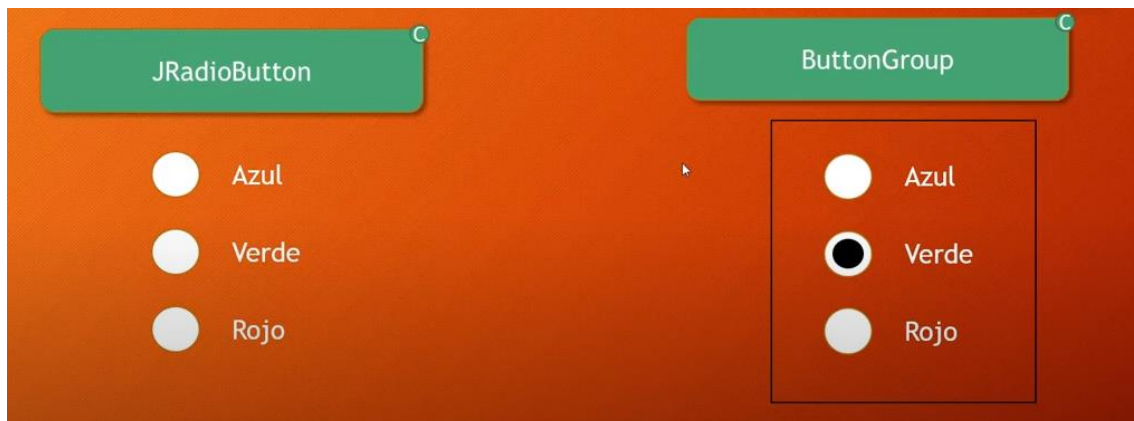
```





Componentes Swing. Botones de radio. (Vídeo 93)

Botones de radio (JRadioButton)

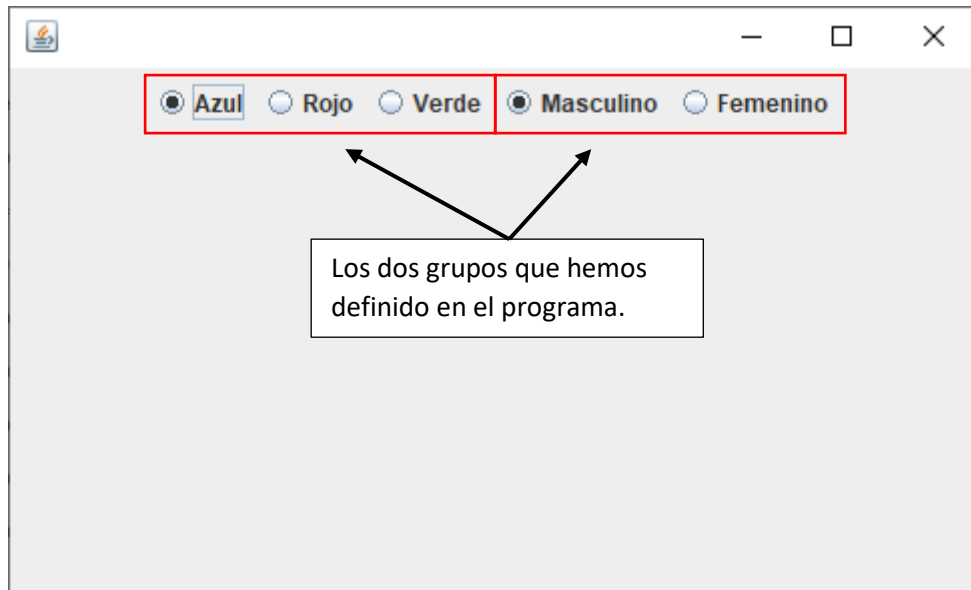


```
1 package graficos;
2 import javax.swing.*;
3 public class Sintaxis_radio {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Marco_radio_sintaxis mimarco=new Marco_radio_sintaxis();
8         mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9     }
10
11 }
12 class Marco_radio_sintaxis extends JFrame{
13     public Marco_radio_sintaxis() {
14         setVisible(true);
15         setBounds(550,300,500,300);
16         Lamina_radio_sintaxis milamina=new Lamina_radio_sintaxis();
17         add(milamina);
18     }
19 }
20
21 class Lamina_radio_sintaxis extends JPanel{
22     public Lamina radio_sintaxis() {
23         ButtonGroup migrupol=new ButtonGroup();
24         JRadioButton boton1=new JRadioButton("Azul",false);
25         JRadioButton boton2=new JRadioButton("Rojo",true);
26         JRadioButton boton3=new JRadioButton("Verde",false);
27         migrupol.add(boton1);
28         migrupol.add(boton2);
29         migrupol.add(boton3);
30         add(boton1);
31         add(boton2);
32         add(boton3);
33         ButtonGroup migrupol2=new ButtonGroup();
34         JRadioButton boton4=new JRadioButton("Masculino",false);
35         JRadioButton boton5=new JRadioButton("Femenino",false);
36         migrupol2.add(boton4);
37         migrupol2.add(boton5);
38         add(boton4);
39         add(boton5);
40     }
41 }
```

Se agregan los botones,
no el grupo.

Se agregan los botones,
no el grupo.

Este será el resultado:



Componentes Swing. Botones de radio II. (Vídeo 94)

```
package graficos;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Ejemplo_radio {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Marco_radio mimarco=new Marco_radio();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class Marco_radio extends JFrame{
    public Marco_radio() {
        setVisible(true);
        setTitle("Ventana con botones radio");
        setBounds(550,300,500,350);
        Lamina_radio milamina=new Lamina_radio();
        add(milamina);
    }
}

class Lamina_radio extends JPanel{
    public Lamina_radio() {
        setLayout(new BorderLayout());
        texto=new JLabel("En un lugar de la Mancha de cuyo nombre...");
        add(texto, BorderLayout.CENTER);
        ButtonGroup migruppo=new ButtonGroup();
        boton1=new JRadioButton("Pequeño", false);
        boton2=new JRadioButton("Mediano", true);
        boton3=new JRadioButton("Grande", false);
        boton4=new JRadioButton("Muy Grande", false);
        JPanel lamina_radio=new JPanel();
        evento_radio mievento=new evento_radio();
        boton1.addActionListener(mievento);
        boton2.addActionListener(mievento);
        boton3.addActionListener(mievento);
        boton4.addActionListener(mievento);
        migruppo.add(boton1);
        migruppo.add(boton2);
        migruppo.add(boton3);
        migruppo.add(boton4);
        lamina_radio.add(boton1);
        lamina_radio.add(boton2);
        lamina_radio.add(boton3);
        lamina_radio.add(boton4);
        add(lamina_radio, BorderLayout.SOUTH);
    }
    private class evento_radio implements ActionListener{
```

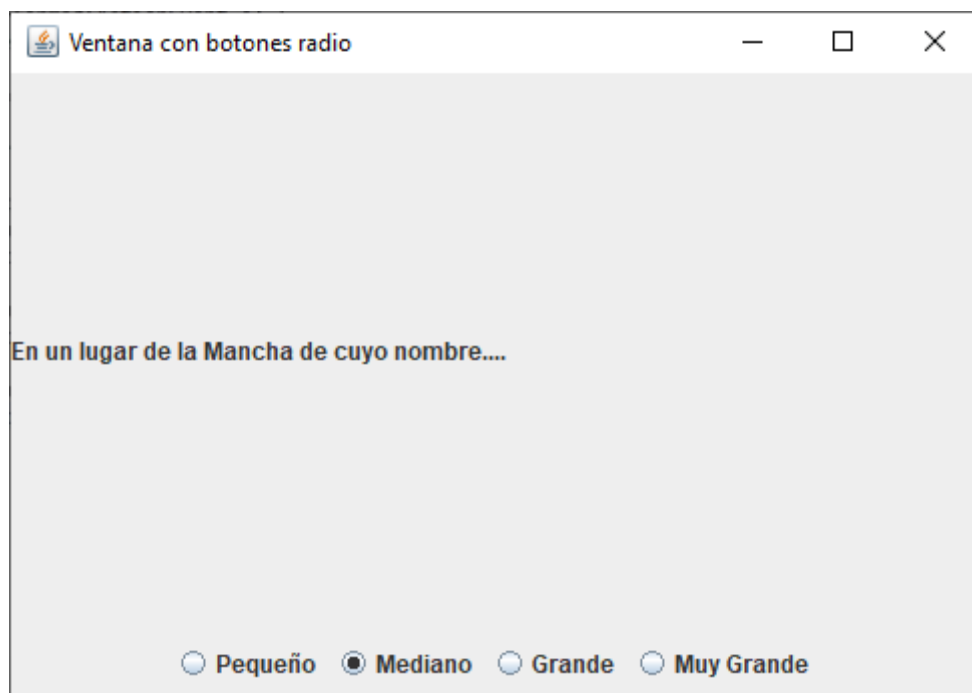
```

@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if(e.getSource()==boton1) {
        texto.setFont(new Font("Serif",Font.PLAIN,10));
    }
    else if(e.getSource()==boton2) {
        texto.setFont(new Font("Serif",Font.PLAIN,12));
    }
    else if(e.getSource()==boton3) {
        texto.setFont(new Font("Serif",Font.PLAIN,18));
    }
    else if(e.getSource()==boton4) {
        texto.setFont(new Font("Serif",Font.PLAIN,24));
    }
}

private JLabel texto;
private JRadioButton boton1, boton2, boton3, boton4;
}

```

Este será el resultado:



Ahora puedes probar los distintos tamaños de fuente.

Ahora vamos a modificar el código anterior para poder disminuir las líneas de código.

```

package graficos;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Ejemplo_radio {

```

```

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Marco_radio mimarco=new Marco_radio();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class Marco_radio extends JFrame{
    public Marco_radio() {
        setVisible(true);
        setTitle("Ventana con botones radio");
        setBounds(550,300,500,350);
        Lamina_radio milamina=new Lamina_radio();
        add(milamina);
    }
}

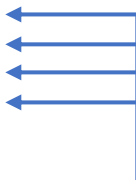
class Lamina_radio extends JPanel{
    public Lamina_radio() {
        setLayout(new BorderLayout());
        texto=new JLabel("En un lugar de la Mancha de cuyo nombre...");
        texto.setFont(new Font("Serif",Font.PLAIN,12));
        add(texto, BorderLayout.CENTER);
        lamina_botones=new JPanel();
        migruo=new ButtonGroup();
        colocarBotones("Pequeño", false, 10);
        colocarBotones("Mediano", true, 12);
        colocarBotones("Grande", false, 18);
        colocarBotones("Muy Grande", false, 24);
        add(lamina_botones, BorderLayout.SOUTH);
    }
}

    public void colocarBotones(String nombre, boolean seleccionado, final
int tamagno) {
        JRadioButton boton=new JRadioButton(nombre, seleccionado);
        migruo.add(boton);
        lamina_botones.add(boton);
        ActionListener mievento=new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                texto.setFont (new Font("Serif", Font.PLAIN,
tamagno));
            }
        };
        boton.addActionListener(mievento);
    }

private JLabel texto;
private JRadioButton boton1, boton2, boton3, boton4;
private ButtonGroup migruo;
private JPanel lamina_botones;
}

```



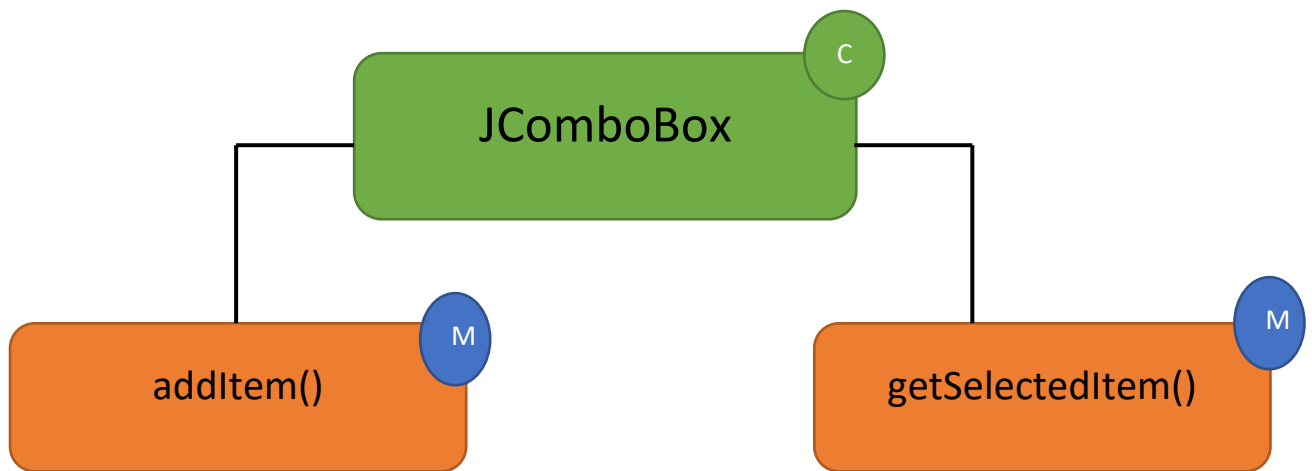
El método colocarBotones habría que repetirlo 4 veces de este modo solo se escribe una vez y es llamado cuatro veces con sus parámetros respectivos, el resultado es el mismo que en código anterior.

A slide with a dark orange background featuring a world map. The slide contains the following text and logos:

- Top left: A small Java logo.
- Top center: **CURSO JAVA** in large white letters.
- Top right: The number **94** in white on a yellow square background.
- Center: **COMPONENTES SWING** in white.
- Below center: **BOTONES DE RADIO II (JRadioButton)** in white.
- Bottom left: The word **eclipse** in a light, lowercase font.
- Bottom right: The Java logo (a blue coffee cup with orange steam) and the word **Java™** in orange.

Componentes Swing. ComboBox. (VÍdeo 95)

Menús desplegables.



```
public class PruebaCombo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoCombo mimarco=new MarcoCombo();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}

class MarcoCombo extends JFrame{
    public MarcoCombo() {
        setVisible(true);
        setBounds(550,300,550,400);
        LaminaCombo milamina=new LaminaCombo();
        add(milamina);
    }
}

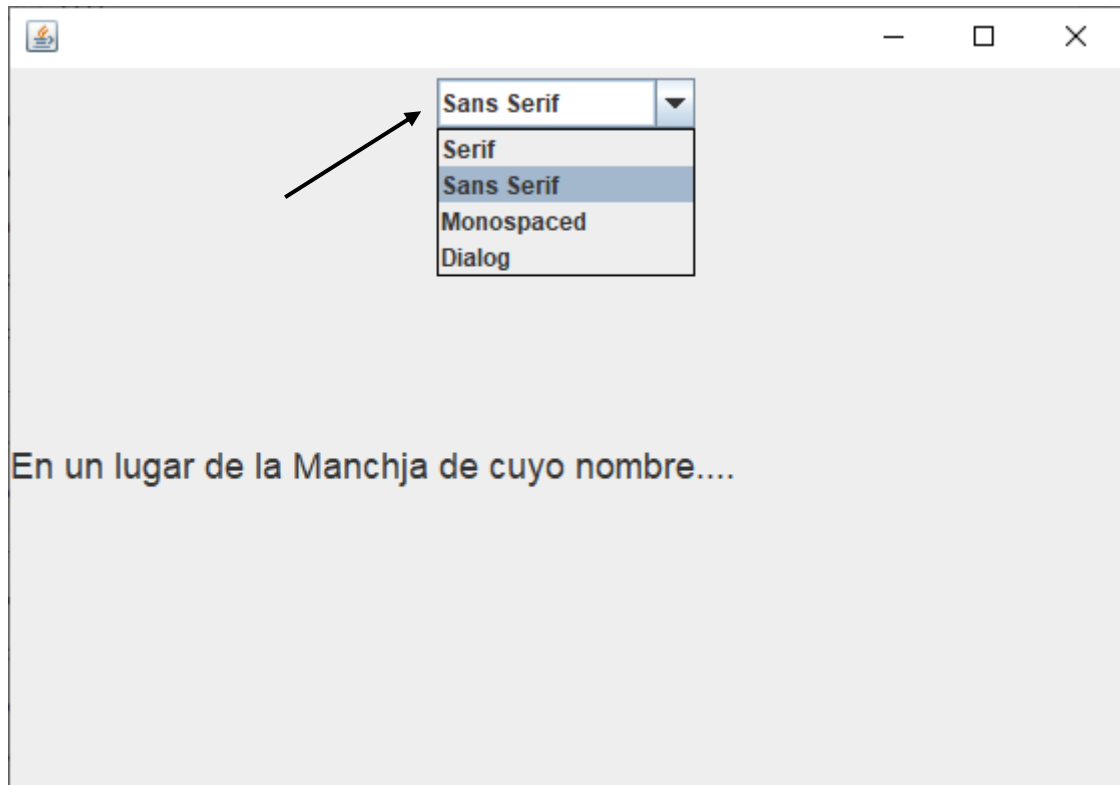
class LaminaCombo extends JPanel{
    public LaminaCombo() {
        setLayout(new BorderLayout());
        texto=new JLabel("En un lugar de la Manchja de cuyo
nombre...");
        texto.setFont(new Font("Serif", Font.PLAIN, 18));
        add(texto, BorderLayout.CENTER);
        JPanel lamina_norte=new JPanel();
        micombo=new JComboBox();
        micombo.setEditable(true);
        micombo.addItem("Serif");
        micombo.addItem("Sans Serif");
        micombo.addItem("Monospaced");
        micombo.addItem("Dialog");
        Evento_combo mievento=new Evento_combo();
        micombo.addActionListener(mievento);

        lamina_norte.add(micombo);
        add(lamina_norte, BorderLayout.NORTH);
    }
    private class Evento_combo implements ActionListener{
```

```
@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    texto.setFont(new Font((String)micombo.getSelectedItem(),
Font.PLAIN, 18));
}

private JLabel texto;
private JComboBox micombo;
}
```

Este será el resultado:

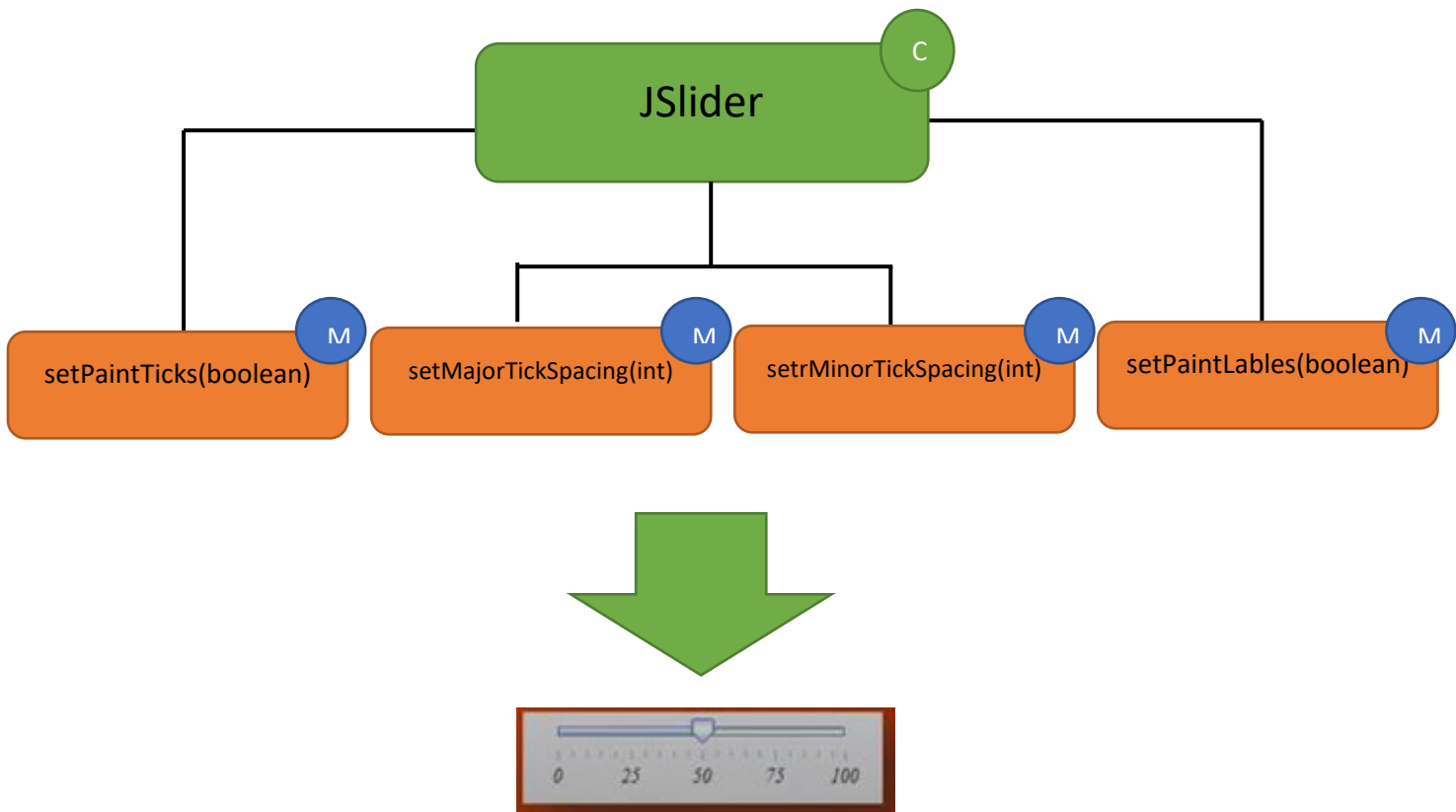


Además puedes introducir por teclas otras opciones.





Componentes Swing. JSlider I. (VÍdeo 96)



```
package graficos;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MarcoSliders {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Frame_Sliders mimarco=new Frame_Sliders();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}

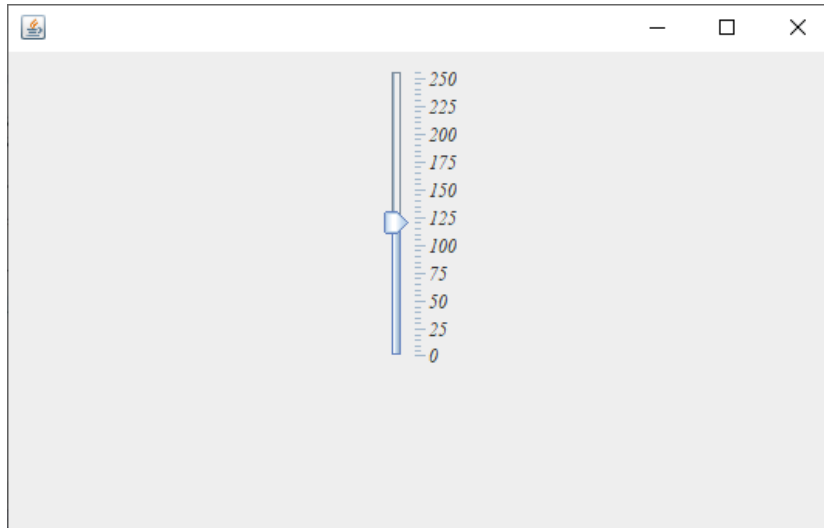
class Frame_Sliders extends JFrame{
    public Frame_Sliders() {
        setBounds(550,400,550,350);
        Lamina_Sliders milamina=new Lamina_Sliders();
        add(milamina);
        setVisible(true);
    }
}

class Lamina_Sliders extends JPanel{
    public Lamina_Sliders() {

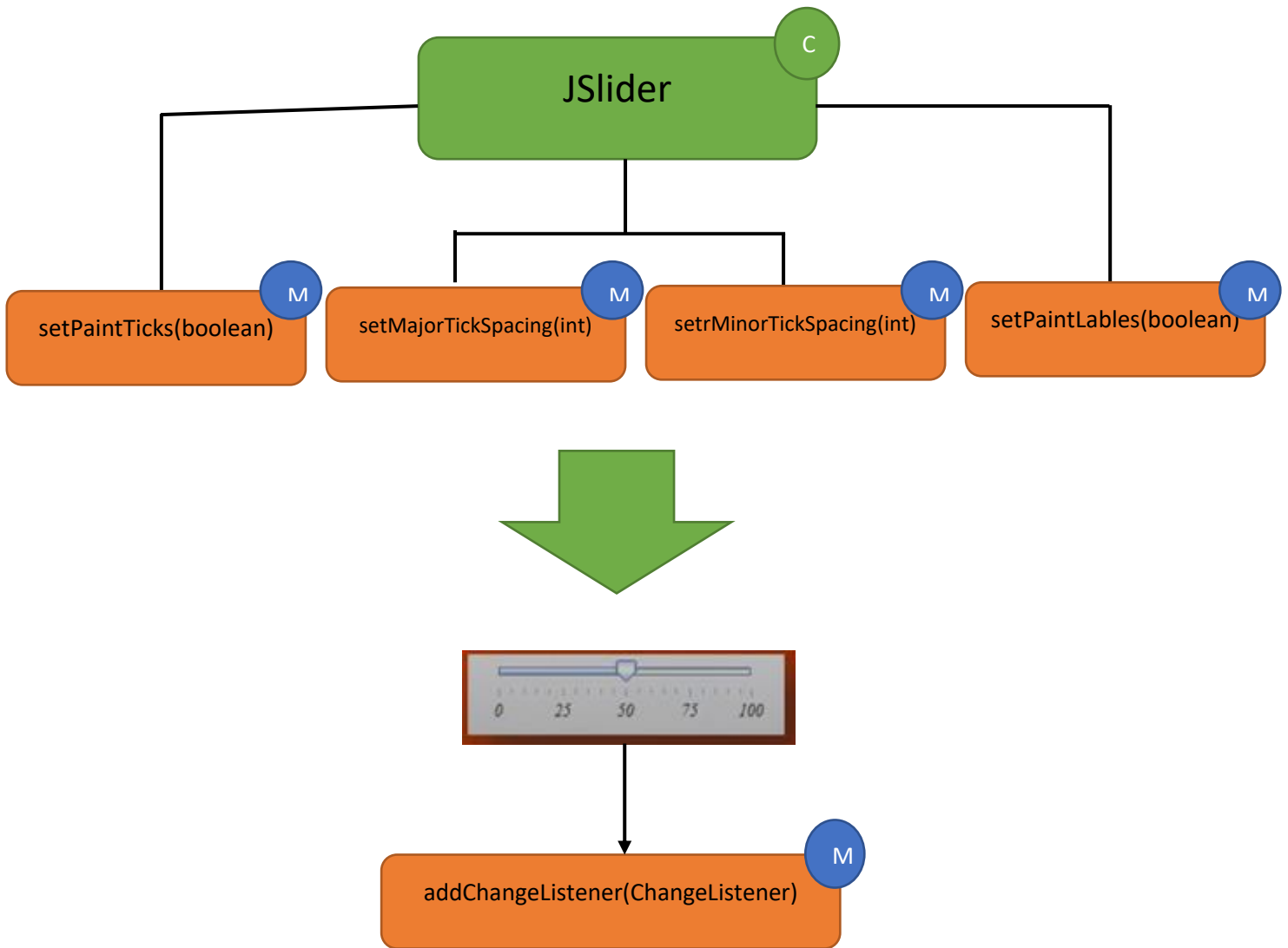
        //JSlider control=new
        JSlider(SwingConstants.VERTICAL,0,255,120);
        JSlider control=new JSlider(0,255,120);
        control.setOrientation(SwingConstants.VERTICAL);//Orientación
        vertical
    }
}
```

```
control.setMajorTickSpacing(25); //Cada 25 marcar grande
control.setMinorTickSpacing(5); //Cada 5 marca pequeña
control.setPaintTicks(true); //Dibuja pequeñas líneas
control.setFont(new Font("Serif", Font.ITALIC, 12));
control.setPaintLabels(true); //Pone numeración en la barra
control.setSnapToTicks(true); //Se ajusta a la marca siguiente
add(control);
}
```

Este será el resultado:



Componentes Swing. JSlider II. (VÍdeo 97)



```
package graficos;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.awt.event.*;

public class MarcoSliders {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Frame_Sliders mimarco=new Frame_Sliders();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

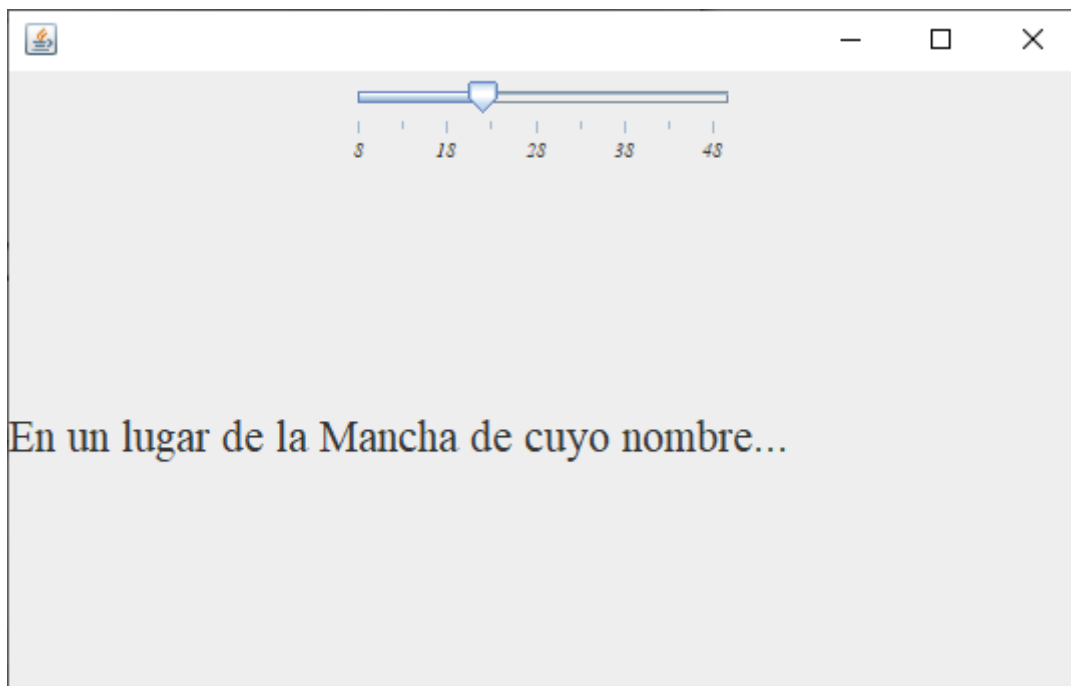
class Frame_Sliders extends JFrame{
    public Frame_Sliders() {
        setBounds(550,400,550,350);
```

```

        Lamina_Sliders milamina=new Lamina_Sliders();
        add(milamina);
        setVisible(true);
    }
}
class Lamina_Sliders extends JPanel{
    public Lamina_Sliders() {
        setLayout(new BorderLayout());
        rotulo=new JLabel("En un lugar de la Mancha de cuyo nombre...");
        add(rotulo, BorderLayout.CENTER);
        control = new JSlider(8,50,12);
        control.setMajorTickSpacing(10);
        control.setMinorTickSpacing(5);
        control.setPaintTicks(true);
        control.setPaintLabels(true);
        control.setFont(new Font("Serif", Font.ITALIC,10));
        control.addChangeListener(new EventoSlider());
        JPanel laminaSlider=new JPanel();
        laminaSlider.add(control);
        add(laminaSlider, BorderLayout.NORTH);
    }
    private class EventoSlider implements ChangeListener{
        @Override
        public void stateChanged(ChangeEvent e){
            // TODO Auto-generated method stub
            //System.out.println("Estás manipulando el deslizando al
valor " + control.getValue());
            rotulo.setFont(new Font("Serif",Font.PLAIN,
control.getValue()));
        }
    }
    private JLabel rotulo;
    private JSlider control;
}

```

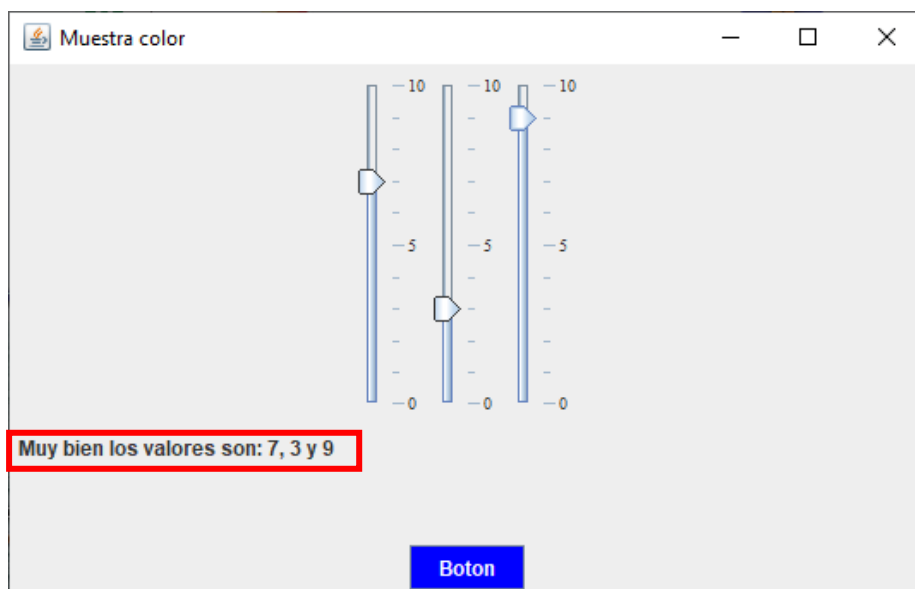
Este será el resultado:



Según vayamos moviendo la barra deslizadora cambiará el tamaño del texto.

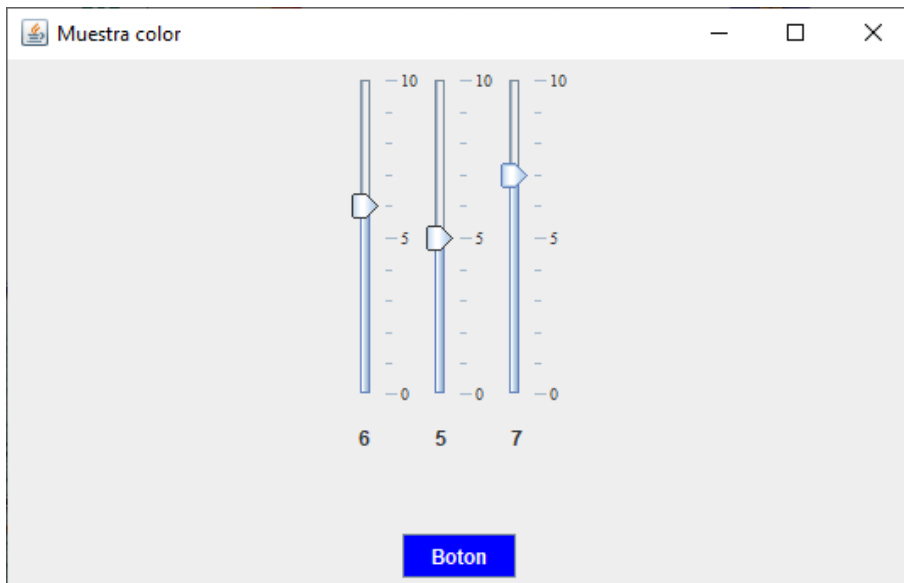


Muy bien ahora para repasar lo aprendido hasta ahora vamos a realizar el siguiente ejercicio.

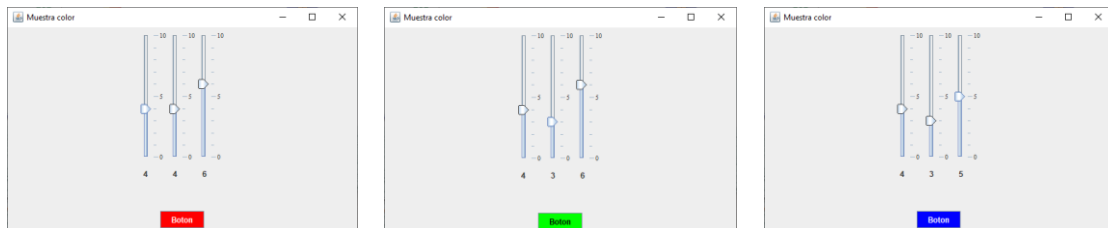


Consiste en hacer el siguiente proyecto, si aciertas la combinación en los siguientes Sliders que aparezca el siguiente mensaje “Muy bien los valores son: 7, 3 y 9”.

Si esta otros valores esta será la ventana:



Además si mueves la primera barra el botón se vuelve de fondo rojo, el segundo se vuelve de fondo verde y el tercero de fondo azul con letras blancas.



En la siguiente pagina te adjunto el código:

```

package graficos;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;
import javax.swing.event.*;
public class ColorFondo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Frame_color mimarco=new Frame_color();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    class Frame_color extends JFrame{
    public Frame_color() {

        //this.getContentPane().setBackground(Color.RED);

        setBounds(550,400,550,350);
        setTitle("Muestra color");

        Laminacolor milamina=new Laminacolor();
        add(milamina);

        setVisible(true);

    }
}

class Laminacolor extends JPanel{
    public Laminacolor() {

        setLayout(new BorderLayout());

        control1=new JSlider(0,10,0);
        control1.setOrientation(SwingConstants.VERTICAL);
        control1.setMajorTickSpacing(5);
        control1.setMinorTickSpacing(1);
        control1.setPaintTicks(true);
        control1.setPaintLabels(true);
        control1.setFont(new Font("Serif", Font.PLAIN, 10));
        control1.setPaintLabels(true);

        control2=new JSlider(0,10,0);
        control2.setOrientation(SwingConstants.VERTICAL);
        control2.setMajorTickSpacing(5);
        control2.setMinorTickSpacing(1);
        control2.setPaintTicks(true);
        control2.setPaintLabels(true);
        control2.setFont(new Font("Serif", Font.PLAIN, 10));
        control2.setPaintLabels(true);

        control3=new JSlider(0,10,0);
        control3.setOrientation(SwingConstants.VERTICAL);
        control3.setMajorTickSpacing(5);

```

```

control3.setMinorTickSpacing(1);
control3.setPaintTicks(true);
control3.setPaintLabels(true);
control3.setFont(new Font("Serif", Font.PLAIN, 10));
control3.setPaintLabels(true);

JPanel laminabarras=new JPanel();

laminabarras.add(control1);
laminabarras.add(control2);
laminabarras.add(control3);

control1.addChangeListener(new ejecuta());
control2.addChangeListener(new ejecuta());
control3.addChangeListener(new ejecuta());

add(laminabarras, BorderLayout.NORTH);

eti1=new JLabel("  0      ");
eti2=new JLabel("  0      ");
eti3=new JLabel("  0      ");

JPanel Laminacolores=new JPanel();

Laminacolores.add(eti1);
Laminacolores.add(eti2);
Laminacolores.add(eti3);

add(Laminacolores, BorderLayout.CENTER);

JPanel laminapie=new JPanel();

boton=new JButton("Boton");
boton.addActionListener(new clic());
laminapie.add(boton);
add(laminapie, BorderLayout.SOUTH);

JPanel Laminaizq=new JPanel();
textoFinal=new JLabel("Muy bien los valores son: 7, 3 y 9");
textoFinal.setVisible(false);
Laminaizq.add(textoFinal);
add(Laminaizq, BorderLayout.WEST);
}
private class clic implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        control1.setValue(0);
        control2.setValue(0);
        control3.setValue(0);
    }
}

private class ejecuta implements ChangeListener{

```

```

@Override
public void stateChanged(ChangeEvent e) {
    // TODO Auto-generated method stub
    eti1.setText(Integer.toString(control1.getValue())+"
");
    eti2.setText(Integer.toString(control2.getValue())+"
");
    eti3.setText(Integer.toString(control3.getValue())+"
");

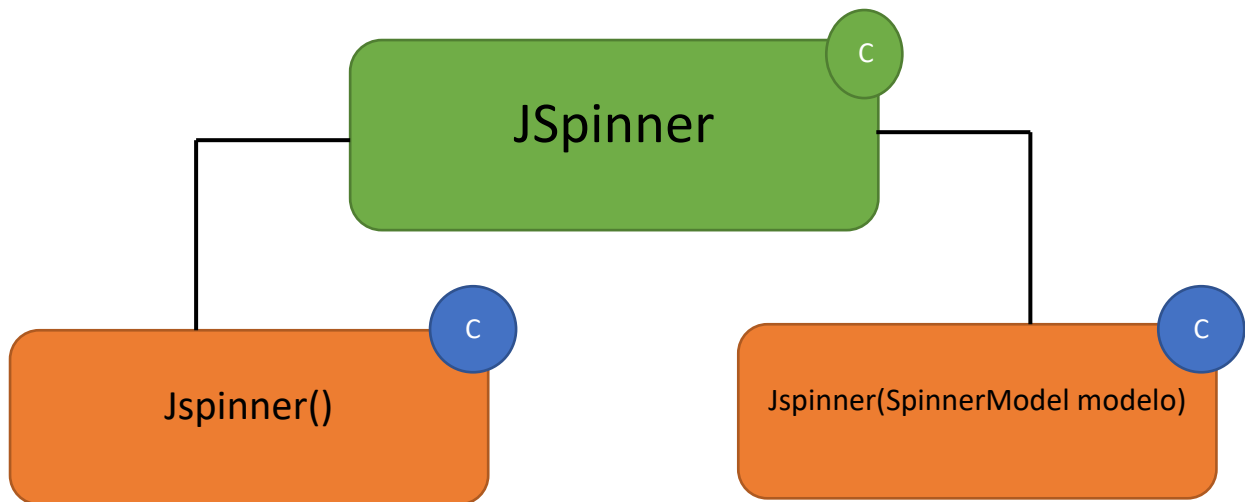
    if(control1.getValueIsAdjusting()) {
        boton.setBackground(Color.RED);
        boton.setForeground(Color.white);
    }
    if(control2.getValueIsAdjusting()) {
        boton.setBackground(Color.GREEN);
        boton.setForeground(Color.black);
    }
    if(control3.getValueIsAdjusting()) {
        boton.setBackground(Color.BLUE);
        boton.setForeground(Color.white);
    }

    if(control1.getValue()==7 && control2.getValue()==3 &&
control3.getValue()==9 ) {
        textoFinal.setVisible(true);
        eti1.setVisible(false);
        eti2.setVisible(false);
        eti3.setVisible(false);
    }
    else {
        textoFinal.setVisible(false);
        eti1.setVisible(true);
        eti2.setVisible(true);
        eti3.setVisible(true);
    }
}

private JSlider control1, control2, control3;
private JLabel eti1, eti2, eti3, textoFinal;
private JButton boton;
private Color micolor;
}

```

Componentes Swing. JSpinner I. (Vídeo 98)



```
1 package graficos;
2
3 import javax.swing.*;
4
5 public class MarcoSpinner {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         FrameSpinner mimarco = new FrameSpinner();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        mimarco.setVisible(true);
12    }
13
14 }
15
16 class FrameSpinner extends JFrame{
17     public FrameSpinner() {
18         setBounds(550,350,550,350);
19         setVisible(true);
20         add(new LaminaSpinner());
21     }
22 }
23
24
25 class LaminaSpinner extends JPanel{
26     public LaminaSpinner(){
27         JSpinner control=new JSpinner();
28         add(control);
29     }
30 }
31 }
```

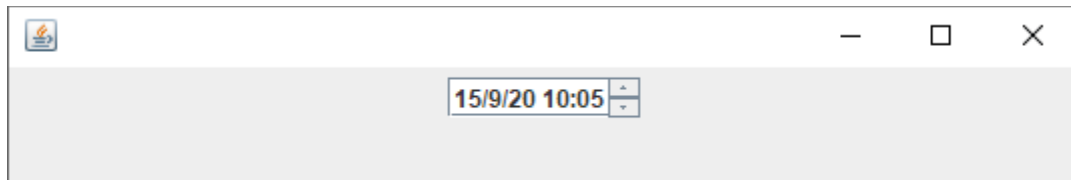
Este será el resultado:



Podemos utilizar los modelos `SpinnerDateModel`, `SpinnerListModel` y `SpinnerNumberModel` para números con un formato específico.

```
25 class LaminaSpinner extends JPanel{
26     public LaminaSpinner(){
27         JSpinner control=new JSpinner(new SpinnerDateModel());
28         add(control);
29     }
```

Este será el resultado:



```
25 class LaminaSpinner extends JPanel{
26     public LaminaSpinner(){
27         String lista[]= {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
28             "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"};
29         JSpinner control=new JSpinner(new SpinnerListModel(lista));
30         add(control);
31     }
32 }
```

Este será el resultado:



Como dimensión coge el primer valor que es Enero, cuando seleccionamos un mes que tiene más caracteres este es el resultado, para arreglarlo vamos a realizar lo siguiente.

```
27 class LaminaSpinner extends JPanel{
28     public LaminaSpinner(){
29         String lista[]= {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
30             "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"};
31         JSpinner control=new JSpinner(new SpinnerListModel(lista));
32         Dimension d=new Dimension(90,20);
33         control.setPreferredSize(d);
34         add(control);
35     }
36 }
```

Este será el resultado:



Hay que importar:

```
import java.awt.Dimension;
```

```

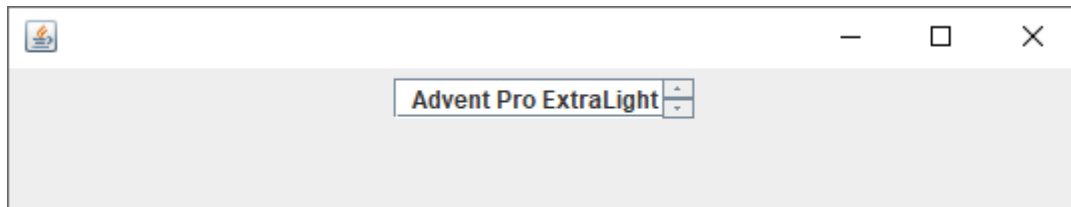
28 class LaminaSpinner extends JPanel{
29     public LaminaSpinner(){
30         String lista[]= GraphicsEnvironment.getLocalGraphicsEnvironment()
31             .getAvailableFontFamilyNames();
32         JSpinner control=new JSpinner(new SpinnerListModel(lista));
33         Dimension d=new Dimension(150,20);
34         control.setPreferredSize(d);
35         add(control);
36     }
37 }

```

Si queremos que nos salga todas los tipos de letras que tenemos instalado en el ordenador, en la lista almacenaremos:

```
GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyNames();
```

Este será el resultado:



Hay que importar:

```
import java.awt.GraphicsEnvironment;
```

Si queremos controla valor mínimo, valor máximo, incremento, etc.

```

28 class LaminaSpinner extends JPanel{
29     public LaminaSpinner(){
30
31         JSpinner control=new JSpinner(new SpinnerNumberModel(5,0,10,1));
32         Dimension d=new Dimension(50,20);
33         control.setPreferredSize(d);
34         add(control);
35     }
36 }

```

El primer parámetro que es el 5 es el valor inicial, el 0 el valor mínimo, el 10 el valor máximo y el 1 el incremento.

Estos valores los podremos modificar a nuestras necesidades.

Este será el resultado:



The image shows a course cover with a dark orange background. At the top left is a small Java logo. The main title 'CURSO JAVA' is in large white letters. To the right, the number '98' is in a yellow box. Below the title, 'COMPONENTES SWING' and 'JSPINNER I' are written in white. The Java logo is on the right, and the Eclipse logo is on the left. A world map is faintly visible in the background.

CURSO JAVA 98

COMPONENTES SWING
JSPINNER I

eclipse Java™

Componentes Swing. Jspinner II. (VÍdeo 99)

```
1 package graficos;
2 import java.awt.Dimension;
3 import java.awt.GraphicsEnvironment;
4 import javax.swing.*;
5
6 public class MarcoSpinner {
7
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10        FrameSpinner mimarco = new FrameSpinner();
11        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12        mimarco.setVisible(true);
13    }
14 }
15 class FrameSpinner extends JFrame{
16     public FrameSpinner() {
17         setBounds(550,350,550,350);
18         setVisible(true);
19         add(new LaminaSpinner());
20     }
21 }
22 class LaminaSpinner extends JPanel{
23     public LaminaSpinner(){
24
25         JSpinner control=new JSpinner(new MiModeloJspinner());
26         Dimension d=new Dimension(50,20);
27         control.setPreferredSize(d);
28         add(control);
29     }
30     private class MiModeloJspinner extends SpinnerNumberModel{
31     public MiModeloJspinner() {
32         super(5,0,10,1);
33     }
34     public Object getNextValue() {
35         return super.getPreviousValue();
36     }
37     public Object getPreviousValue() {
38         return super.getNextValue();
39     }
40 }
41 }
```

Desde la línea 30 hasta la 39 creamos una clase interna llamada MiModeloJspinner que hereda de SpinnerNumberModel.

A la clase que hereda le pasamos los parámetros 5,0,10,1 con super.

Sobrescribamos los métodos getNextValue y getPreviousValue diciéndoles que hagan justamente lo contrario que retornamos al método SpinnerNumberModel con la opción super.

En la línea 25 pasamos al nuevo MiModeloJspinner.

Clases internas anónimas:

```
1 package graficos;
2 import java.awt.Dimension;
3 import java.awt.GraphicsEnvironment;
4 import javax.swing.*;
5
6 public class MarcoSpinner {
7
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10        FrameSpinner mimarco = new FrameSpinner();
11        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12        mimarco.setVisible(true);
13    }
14 }
15 class FrameSpinner extends JFrame{
16     public FrameSpinner() {
17         setBounds(550,350,550,350);
18         setVisible(true);
19         add(new LaminaSpinner());
20     }
21 }
22 class LaminaSpinner extends JPanel{
23     public LaminaSpinner(){
24
25         JSpinner control=new JSpinner(new SpinnerNumberModel(5,0,10,1) {
26             public Object getNextValue() {
27                 return super.getPreviousValue();
28             }
29             public Object getPreviousValue() {
30                 return super.getNextValue();
31             }
32         });
33         Dimension d=new Dimension(50,20);
34         control.setPreferredSize(d);
35         add(control);
36     }
37 }
38 }
39
```

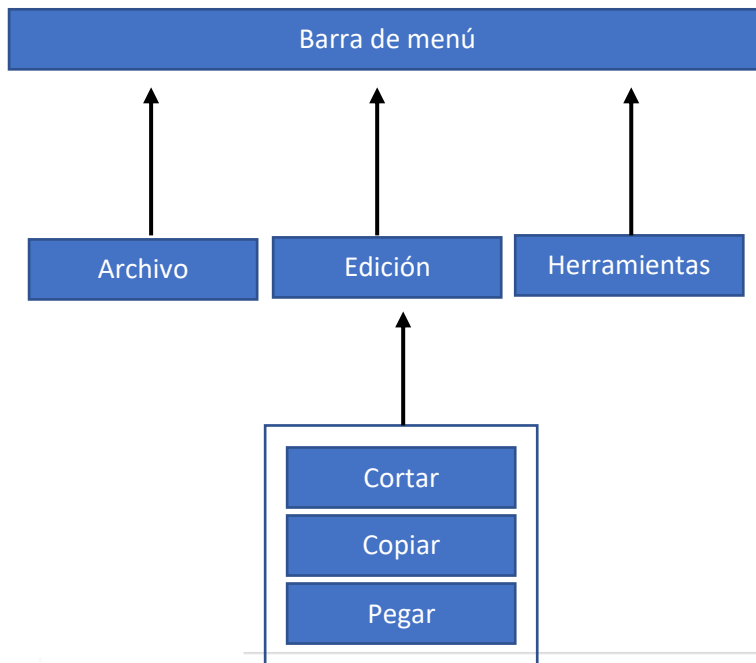
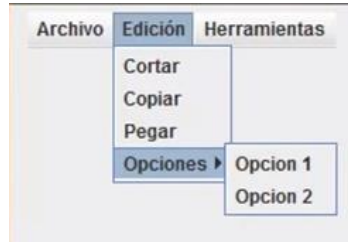
Después de definir el objeto con sus parámetros, abrimos llave e introducimos el código que hace que invirtamos los botones de aumentar y disminuir, se dice anónima porque no se define la clase, es interna y ahorra código, puedes comparar con el código anterior.





Componentes Swing. Creación de menús I. (Vídeo 100)

JMenuBar, JMenu, JMenuItem.



JMenuBar C

JMenu C

JMenuItem C

```
1 package graficos;
2
3 import javax.swing.*;
4
5
6 public class MarcoMenu {
7
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10        MenuFrame mimarco=new MenuFrame();
11        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12    }
13
14 }
15
16 class MenuFrame extends JFrame{
17     public MenuFrame() {
18         setBounds(500,300,550,400);
19         MenuLamina milamina=new MenuLamina();
20         add(milamina);
21         setVisible(true);
22     }
23 }
24
```

```

25 class MenuLamina extends JPanel{
26     public MenuLamina() {
27         JMenuBar mibarra=new JMenuBar();
28         JMenu archivo =new JMenu("Archivo");
29         JMenu edicion =new JMenu("Edición");
30         JMenu Herramientas =new JMenu("Herramientas");
31         mibarra.add(archivo);
32         mibarra.add(edicion);
33         mibarra.add(Herramientas);
34         add(mibarra);
35     }
36 }

```

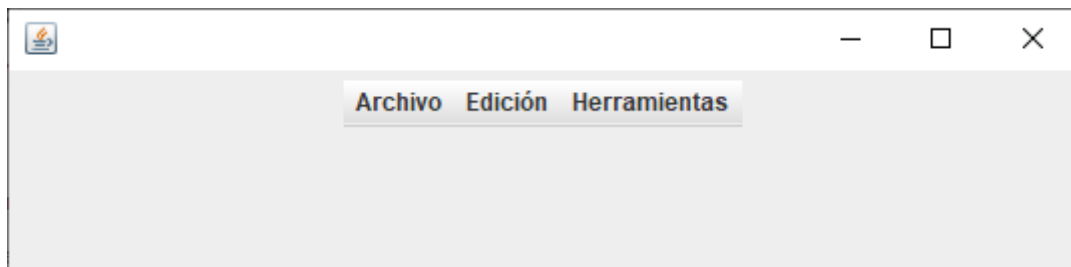
En la línea 27 definimos una barra de menú llamada mibarra.

En las líneas 28 a la 30 definimos las opciones de la barra.

En las líneas 31 a la 33 las opciones las agregamos a la barra de menú.

En la línea 34 agregamos la barra de menú mibarra en el marco, ésta ya será visible desde el marco.

Este será el resultado:



```

24 class MenuLamina extends JPanel{
25     public MenuLamina() {
26         JMenuBar mibarra=new JMenuBar();
27         JMenu archivo =new JMenu("Archivo");
28         JMenu edicion =new JMenu("Edición");
29         JMenu herramientas =new JMenu("Herramientas");
30
31         JMenuItem guardar=new JMenuItem("Guardar");
32         JMenuItem guardar_como=new JMenuItem("Guardar Como...");
33
34         JMenuItem cortar=new JMenuItem("Cortar");
35         JMenuItem copiar=new JMenuItem("Copiar");
36         JMenuItem pegar=new JMenuItem("Pegar");
37
38         JMenuItem generales=new JMenuItem("Generales");
39
40         archivo.add(guardar);
41         archivo.add(guardar_como);
42
43         edicion.add(cortar);
44         edicion.add(copiar);
45         edicion.add(pegar);
46
47         herramientas.add(generales);

```

Guardar y Guardar como los definimos como JMenuItem.

Cortar, copiar y pegar los definimos como JMenuItem.

Generales los definimos como JMenuItem.

```

48
49 mibarra.add(archivo);
50 mibarra.add(edicion);
51 mibarra.add(herramientas);
52 add(mibarra);
53 }

```

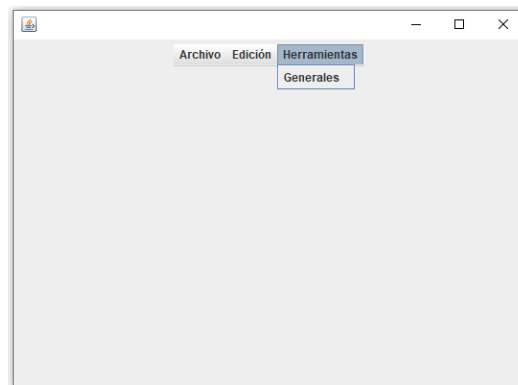
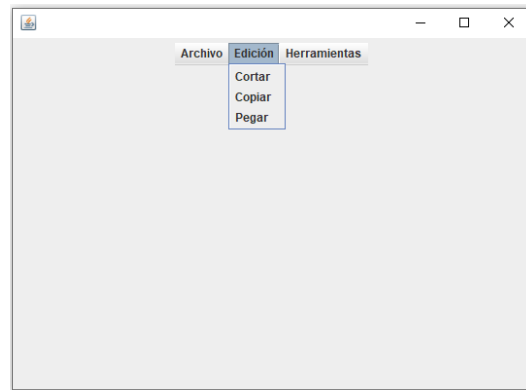
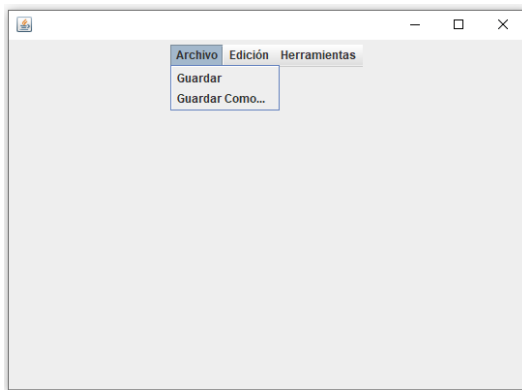
Desde la línea 40 hasta la 47 los agregamos a sus correspondientes menús.

Guardar y Guardar como a Archivo.

Cortar, copiar y pegar a edición.

Generales a herramientas.

Si ejecutamos este será el resultado:



```

24 class MenuLamina extends JPanel{
25     public MenuLamina() {
26         JMenuBar mibarra=new JMenuBar();
27         JMenu archivo =new JMenu("Archivo");
28         JMenu edicion =new JMenu("Edición");
29         JMenu herramientas =new JMenu("Herramientas");
30
31         JMenuItem guardar=new JMenuItem("Guardar");
32         JMenuItem guardar_como=new JMenuItem("Guardar Como...");
33
34         JMenuItem cortar=new JMenuItem("Cortar");
35         JMenuItem copiar=new JMenuItem("Copiar");
36         JMenuItem pegar=new JMenuItem("Pegar");
37         JMenuItem opciones=new JMenu("Opciones");
38         JMenuItem opcion1=new JMenuItem("Opción 1");
39         JMenuItem opcion2=new JMenuItem("Opción 2");
40
41         JMenuItem generales=new JMenuItem("Generales");
42

```

```

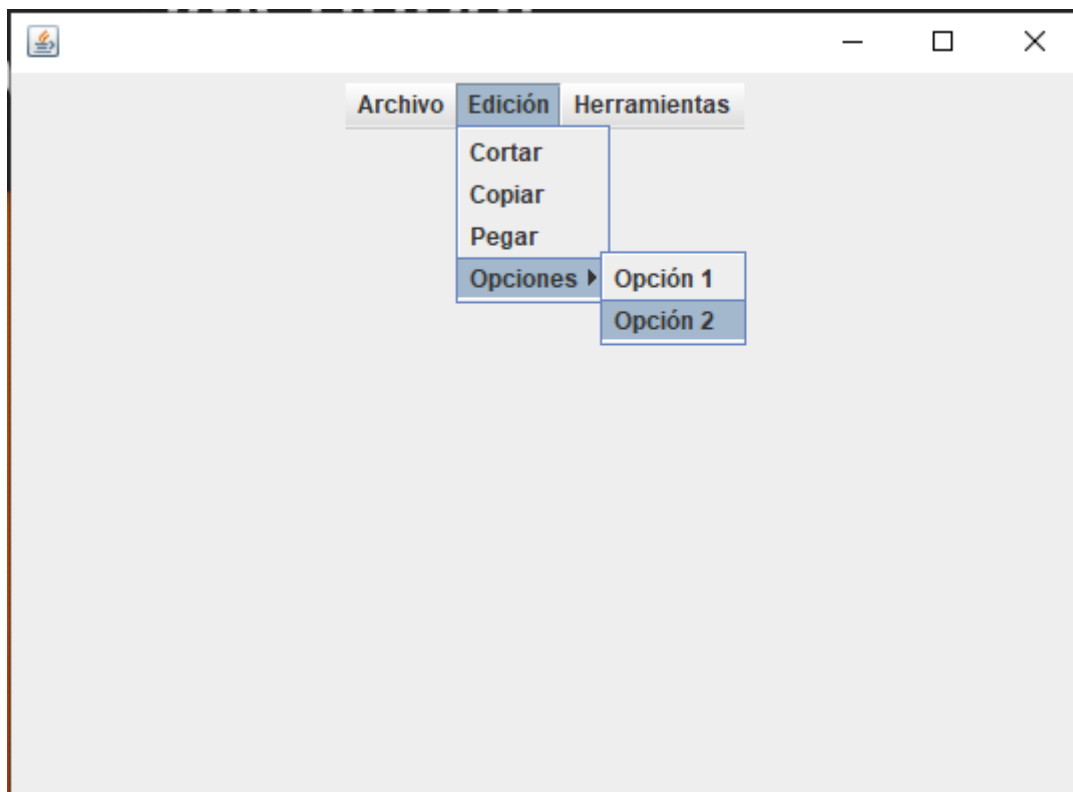
43     archivo.add(guardar);
44     archivo.add(guardar_como);
45
46     edicion.add(cortar);
47     edicion.add(copiar);
48     edicion.add(pegar);
49     edicion.add(opciones);
50
51     opciones.add(opcion1);
52     opciones.add(opcion2);
53
54     herramientas.add(generales);
55
56     mibarra.add(archivo);
57     mibarra.add(edicion);
58     mibarra.add(herramientas);
59     add(mibarra);
60 }

```

Utilizando la misma jerarquía para agregar al menú.

En la línea 35 en lugar de llamar a JMenuItem como este tendrá mas opciones se tiene que llamar JMenu como archivo, edición y herramientas.

Este será el resultado final.



Poner separadores.

```

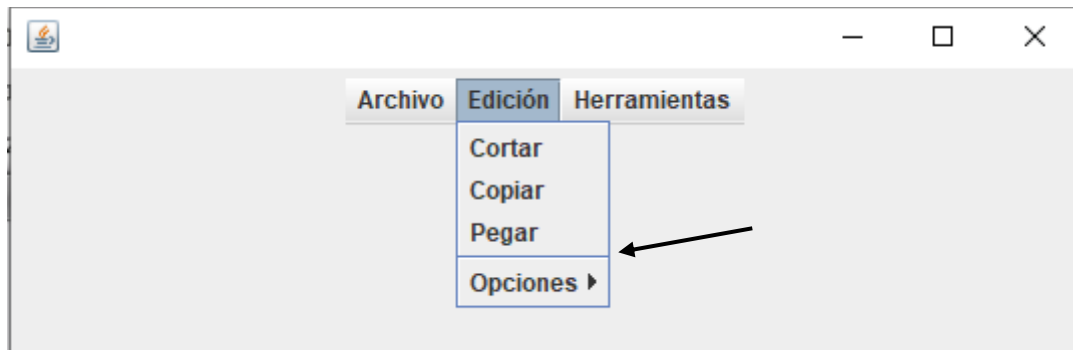
46     edicion.add(cortar);
47     edicion.add(copiar);
48     edicion.add(pegar);
49     edicion.addSeparator(); ←
50     edicion.add(opciones);

```

Nos colocamos donde queremos agregar la separación y escribiremos:

```
edición.addSeparator();
```

Este será el resultado:



Componentes Swing. Creación de procesador de textos. Práctica guiada I. (Vídeo 101)

Con los conocimientos adquiridos vamos a realizar un mini procesador de textos que en un futuro iremos mejorando.



```
package graficos;
import java.awt.BorderLayout;
import java.awt.event.*;
import javax.swing.*;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

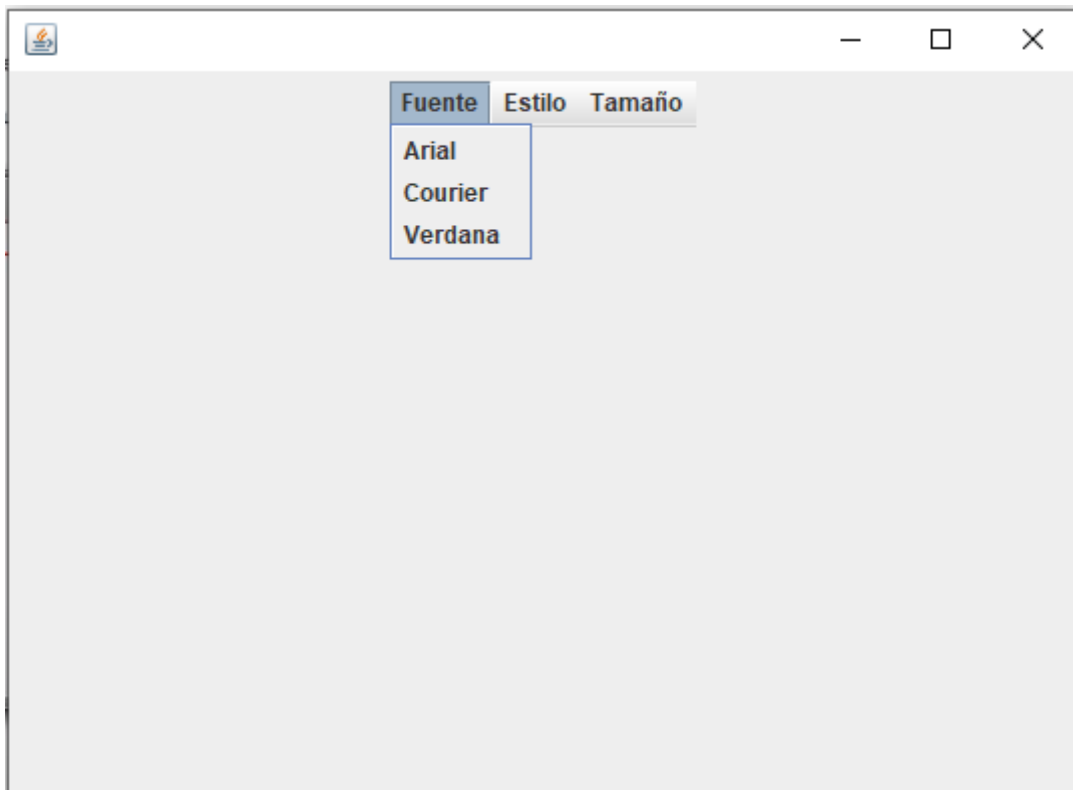
class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        JMenu fuente=new JMenu("Fuente");
        JMenu estilo=new JMenu("Estilo");
        JMenu tamagno=new JMenu("Tamaño");
        //-----
        JMenuItem arial=new JMenuItem("Arial");
        JMenuItem courier=new JMenuItem("Courier");
        JMenuItem verdana=new JMenuItem("Verdana");
    }
}
```

```

    fuente.add(arial);
    fuente.add(courier);
    fuente.add(verdana);
    //-----
    JMenuItem negrita=new JMenuItem("Negrita");
    JMenuItem cursiva=new JMenuItem("Cursiva");
    estilo.add(negrita);
    estilo.add(cursiva);
    //-----
    JMenuItem tam_12=new JMenuItem("12");
    JMenuItem tam_16=new JMenuItem("16");
    JMenuItem tam_20=new JMenuItem("20");
    JMenuItem tam_24=new JMenuItem("24");
    tamagno.add(tam_12);
    tamagno.add(tam_16);
    tamagno.add(tam_20);
    tamagno.add(tam_24);
    //-----
    mibarra.add(fuente);
    mibarra.add(estilo);
    mibarra.add(tamagno);
    //-----
    laminamenu.add(mibarra);
    add(laminamenu, BorderLayout.NORTH);
}
}

```

Ya hemos creado nuestra barra de menú.





Java Componentes Swing. Creación de procesador de textos. Práctica guiada II. (Vídeo 102)

Ahora vamos a utilizar JTextPane para crear el cuadro de texto.

```
57     laminamenu.add(mibarra);
58     add(laminamenu, BorderLayout.NORTH);
59
60     JTextPane miarea=new JTextPane();
61     add(miarea, BorderLayout.CENTER);
62
63 }
64 }
```

Al final de la clase LaminaProcesador_II después del BorderLayout.NORTH, creamos nuestro objeto de tipo JTextPane llamado miarea y a agregamos a la lámina con BorderLayout.CENTER.

Este será el resultado:

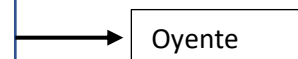


Como primer ejemplo vamos a activar la fuente Courier.

```
class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        JMenu fuente=new JMenu("Fuenta");
        JMenu estilo=new JMenu("Estilo");
        JMenu tamagno=new JMenu("Tamaño");
        //-----
        JMenuItem arial=new JMenuItem("Arial");
        JMenuItem courier=new JMenuItem("Courier");

        Gestiona_menus tipo_letra=new Gestiona_menus();
        courier.addActionListener(tipo_letra);

        JMenuItem verdana=new JMenuItem("Verdana");
        fuente.add(arial);
        fuente.add(courier);
        fuente.add(verdana);
        //-----
        JMenuItem negrita=new JMenuItem("Negrita");
        JMenuItem cursiva=new JMenuItem("Cursiva");
        estilo.add(negrita);
        estilo.add(cursiva);
        //-----
        JMenuItem tam_12=new JMenuItem("12");
```



```

JMenuItem tam_16=new JMenuItem("16");
JMenuItem tam_20=new JMenuItem("20");
JMenuItem tam_24=new JMenuItem("24");
tamagno.add(tam_12);
tamagno.add(tam_16);
tamagno.add(tam_20);
tamagno.add(tam_24);
//-----
mibarra.add(fuente);
mibarra.add(estilo);
mibarra.add(tamagno);
//-----
laminamenu.add(mibarra);
add(laminamenu, BorderLayout.NORTH);

```

```

miarea=new JTextPane();
add(miarea, BorderLayout.CENTER);

```

La declaramos fuera de la clase para que tenga acceso en la clase interna Gestiona_menus.

```

private class Gestiona_menus implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        miarea.setFont(new Font("Courier", Font.PLAIN, 12));
    }
}

```

```

JTextPane miarea;

```

Clase interna ActionListener cuando tenga lugar el evento cambiará la fuente a Courier.

Lo que tenemos en el recuadro oyente también lo podemos simplificar a una línea.

```

Courie.addActionListener(new Gestiona_menus());

```

```

23 class LaminaProcesador_II extends JPanel{
24     public LaminaProcesador_II() {
25         setLayout(new BorderLayout());
26         JPanel laminamenu=new JPanel();
27         JMenuItem mibarra=new JMenuItem();
28         //-----
29         JMenuItem fuente=new JMenuItem("Fuente");
30         JMenuItem estilo=new JMenuItem("Estilo");
31         JMenuItem tamagno=new JMenuItem("Tamaño");
32         //-----
33         JMenuItem arial=new JMenuItem("Arial");
34         JMenuItem courier=new JMenuItem("Courier");
35
36         //Gestiona_menus tipo_letra=new Gestiona_menus();
37         //courier.addActionListener(tipo_letra);
38         courier.addActionListener(new ActionListener() {
39             public void actionPerformed(ActionEvent e) {
40                 // TODO Auto-generated method stub
41                 miarea.setFont(new Font("Courier", Font.PLAIN, 12));
42             }
43         });
44
45

```

Clase interna anónima.

Eliminamos la clase interna Gestiona_menus.

```
23 class LaminaProcesador_II extends JPanel{
24     public LaminaProcesador_II() {
25         setLayout(new BorderLayout());
26         JPanel laminamenu=new JPanel();
27         JMenuBar mibarra=new JMenuBar();
28         //-----
29         JMenu fuente=new JMenu("Fuente");
30         JMenu estilo=new JMenu("Estilo");
31         JMenu tamagno=new JMenu("Tamaño");
32         //-----
33         JMenuItem arial=new JMenuItem("Arial");
34         arial.addActionListener(new ActionListener() {
35             public void actionPerformed(ActionEvent e) {
36                 // TODO Auto-generated method stub
37                 miarea.setFont(new Font("Arial", Font.PLAIN, 12));
38             }
39         });
40         //-----
41         JMenuItem courier=new JMenuItem("Courier");
42
43         //Gestiona_menus tipo_letra=new Gestiona_menus();
44         //courier.addActionListener(tipo_letra);
45         courier.addActionListener(new ActionListener() {
46             public void actionPerformed(ActionEvent e) {
47                 // TODO Auto-generated method stub
48                 miarea.setFont(new Font("Courier", Font.PLAIN, 12));
49             }
50         });
51
52         //-----
53         JMenuItem verdana=new JMenuItem("Verdana");
54         verdana.addActionListener(new ActionListener() {
55             public void actionPerformed(ActionEvent e) {
56                 // TODO Auto-generated method stub
57                 miarea.setFont(new Font("Verdana", Font.PLAIN, 12));
58             }
59         });
60         //-----
```

Hacemos los mismo con Arial y Verdana.

Pero aun nos queda los estilos Negrita y Cursiva, así como los tamaños de 12, 16, 20 y 24, como podrás comprobar aún falta mucho código.

Además de poder combinar el tipo de letra con el estilo y tamaño sería interesante de crear un método que sea el encargado de crear tanto los elementos o de ponerlos a la escucha.





Componentes Swing. Creación de procesador de textos. Práctica guiada III. (Vídeo 103)

Vamos a ver como continuar con el procesador de textos, pero ahora para evitar hacer mucho código cambiaremos el método.

```
package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","", 1,1);
        configura_menu("Courier","fuente","", 1,1);
        configura_menu("Verdama","fuente","", 1,1);

        configura_menu("Negrita","estilo","", 1,1);
        configura_menu("Cursiva","estilo","", 1,1);

        configura_menu("12","tamaño","", 1,1);
        configura_menu("16","tamaño","", 1,1);
        configura_menu("20","tamaño","", 1,1);
        configura_menu("24","tamaño","", 1,1);

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
```

Utilizan el método
configura_menu.


```
add(miarea, BorderLayout.CENTER);  
}
```

```
public void configura_menu(String rotulo, String menu, String  
tipo_letra, int estilos, int tam) {  
    JMenuItem elem_menu=new JMenuItem(rotulo);  
    if(menu=="fuente") {  
        fuente.add(elem_menu);  
    }  
    else if(menu=="estilo") {  
        estilo.add(elem_menu);  
    }  
    else if(menu=="tamaño") {  
        tamagno.add(elem_menu);  
    }  
}
```

```
JTextPane miarea;  
JMenu fuente, estilo, tamagno;  
Font letras;  
}
```

Si ejecutamos observaremos que ya están todos los menús.



Componentes Swing. Creación procesador de textos. Práctica guiada IV. (Vídeo 104)

```
package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10);
        configura_menu("Courier","fuente","Courier", 9,10);
        configura_menu("Verdama","fuente","Verdana", 9,10);

        configura_menu("Negrita","estilo","", Font.BOLD,1);
        configura_menu("Cursiva","estilo","", Font.ITALIC,1);

        configura_menu("12","tamaño","", 9,12);
        configura_menu("16","tamaño","", 9,16);
        configura_menu("20","tamaño","", 9,20);
        configura_menu("24","tamaño","", 9,24);

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
        add(miarea, BorderLayout.CENTER);
    }
}
```

Pasamos los parámetros que queremos que cambien en el procesador de textos.

```

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam) {
        JMenuItem elem_menu=new JMenuItem(rotulo);
        if(menu=="fuente") {
            fuente.add(elem_menu);
        }
        else if(menu=="estilo") {
            estilo.add(elem_menu);
        }
        else if(menu=="tamaño") {
            tamagno.add(elem_menu);
        }
        elem_menu.addActionListener(new Gestiona_Eventos(rotulo,
tipo_letra, estilos, tam));
    }

```

```

private class Gestiona_Eventos implements ActionListener{
    String tipo_texto, menu;
    int estilo_letra, tamagno_letra;
    Gestiona_Eventos(String elemento,String texto2, int estilo2, int
tam_letra){
        tipo_texto=texto2;
        estilo_letra=estilo2;
        tamagno_letra=tam_letra;
        menu=elemento;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        miarea.setFont(new Font(tipo_texto, estilo_letra,
tamagno_letra));
    }
}

```

```

JTextPane miarea;
JMenu fuente, estilo, tamagno;
Font letras;
}

```

Ahora nos queda que cuando cambiemos el tipo de letra, estilo o tamaño este no afecte a los otros parámetros, si la letra está en Arial y tamaño 12 cuando seleccionemos Negrita esta tiene que pasar a negrita manteniendo el tipo Arial y tamaño 12.





Componentes Swing. Creación de procesador de textos. Práctica guiada V. (Vídeo 105)

```
package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10);
        configura_menu("Courier","fuente","Courier", 9,10);
        configura_menu("Verdama","fuente","Verdana", 9,10);

        configura_menu("Negrita","estilo","", Font.BOLD,1);
        configura_menu("Cursiva","estilo","", Font.ITALIC,1);

        configura_menu("12","tamaño","", 9,12);
        configura_menu("16","tamaño","", 9,16);
        configura_menu("20","tamaño","", 9,20);
        configura_menu("24","tamaño","", 9,24);

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
        add(miarea, BorderLayout.CENTER);
    }
}
```

```

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam) {
        JMenuItem elem_menu=new JMenuItem(rotulo);
        if(menu=="fuente") {
            fuente.add(elem_menu);
        }
        else if(menu=="estilo") {
            estilo.add(elem_menu);
        }
        else if(menu=="tamaño") {
            tamagno.add(elem_menu);
        }
        elem_menu.addActionListener(new Gestiona_Eventos(rotulo,
tipo_letra, estilos, tam));
    }

```

```

private class Gestiona_Eventos implements ActionListener{
    String tipo_texto, menu;
    int estilo_letra, tamagno_letra;
    Gestiona_Eventos(String elemento,String texto2, int estilo2, int
tam_letra){
        tipo_texto=texto2;
        estilo_letra=estilo2;
        tamagno_letra=tam_letra;
        menu=elemento;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        letras=miarea.getFont();
        if(menu=="Arial" || menu=="Courier" || menu=="Verdama") {
            estilo_letra=letras.getStyle();
            tamagno_letra=letras.getSize();
        }
        else if(menu=="Negrita" || menu=="Cursiva") {
            tipo_texto=letras.getFontName();
            tamagno_letra=letras.getSize();
        }
        else if(menu=="12" | menu=="16" | menu=="20" | menu=="24") {
            estilo_letra=letras.getStyle();
            tipo_texto=letras.getFontName();
        }
        miarea.setFont(new Font(tipo_texto, estilo_letra,
tamagno_letra));
    }
}

```

```

JTextPane miarea;
JMenu fuente, estilo, tamagno;
Font letras;
}

```

Según el menú que seleccionemos que nos conserve los valores que no queremos modificar y solo cambie la opción que seleccionemos.

```

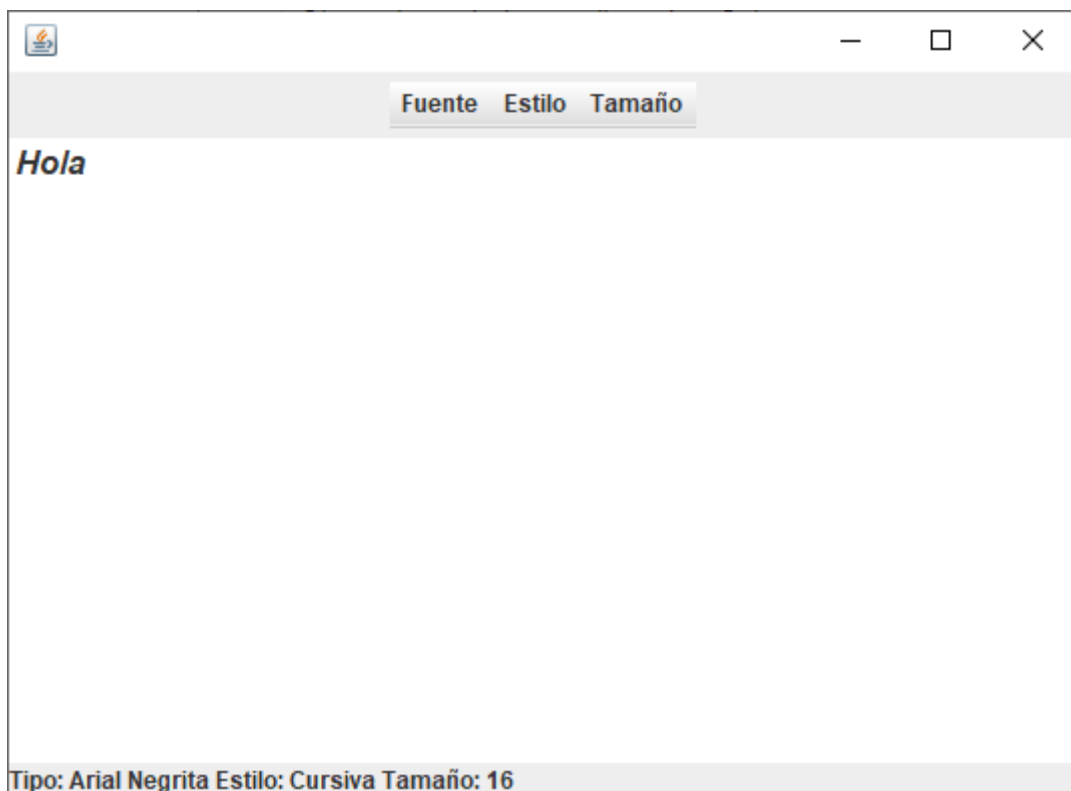
81     public void actionPerformed(ActionEvent e) {
82         // TODO Auto-generated method stub
83         letras=miarea.getFont();
84         if(menu=="Arial" || menu=="Courier" || menu=="Verdama") {
85             estilo_letra=letras.getStyle();
86             tamagno_letra=letras.getSize();
87         }
88         else if(menu=="Negrita" || menu=="Cursiva") {
89             tipo_texto=letras.getFontName();
90             tamagno_letra=letras.getSize();
91         }
92         else if(menu=="12" | menu=="16" | menu=="20" | menu=="24") {
93             estilo_letra=letras.getStyle();
94             tipo_texto=letras.getFontName();
95         }
96         miarea.setFont(new Font(tipo_texto, estilo_letra, tamagno_letra));
97         System.out.println("Tipo: " +tipo_texto +" Estilo: " + estilo_letra + " tamañi: " +tamagno_letra);
98     }
99 }

```

`System.out.println("Tipo: " +tipo_texto +" Estilo: " + estilo_letra + " tamañi: " +tamagno_letra);`

Con esta línea de código nos imprimirá en consola el tipo de letra que tenemos cada vez que hacemos un cambio.

Realiza los pasos necesarios para que aparezca el texto del tipo de letra, estilo y tamaño en la parte inferior de la ventana.



En la siguiente página te adjunto el código:

```

package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import java.awt.font.TextAttribute;

import javax.swing.*;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10);
        configura_menu("Courier","fuente","Courier", 9,10);
        configura_menu("Verdana","fuente","Verdana", 9,10);

        configura_menu("Negrita","estilo","", Font.BOLD,1);
        configura_menu("Cursiva","estilo","", Font.ITALIC,1);

        configura_menu("12","tamaño","", 9,12);
        configura_menu("16","tamaño","", 9,16);
        configura_menu("20","tamaño","", 9,20);
        configura_menu("24","tamaño","", 9,24);

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
        add(miarea, BorderLayout.CENTER);

        informa=new JLabel();
        add(informa, BorderLayout.SOUTH);
    }
}

```



```

    }

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam) {
        JMenuItem elem_menu=new JMenuItem(rotulo);
        if(menu=="fuente") {
            fuente.add(elem_menu);
        }
        else if(menu=="estilo") {
            estilo.add(elem_menu);
        }
        else if(menu=="tamaño") {
            tamagno.add(elem_menu);
        }
        elem_menu.addActionListener(new Gestiona_Eventos(rotulo,
tipo_letra, estilos, tam));
    }

    private class Gestiona_Eventos implements ActionListener{
        String tipo_texto, menu;
        int estilo_letra, tamagno_letra;
        Gestiona_Eventos(String elemento,String texto2, int estilo2, int
tam_letra){
            tipo_texto=texto2;
            estilo_letra=estilo2;
            tamagno_letra=tam_letra;
            menu=elemento;
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            letras=miarea.getFont();
            if(menu=="Arial" || menu=="Courier" || menu=="Verdana") {
                estilo_letra=letras.getStyle();
                tamagno_letra=letras.getSize();
            }
            else if(menu=="Negrita" || menu=="Cursiva") {
                tipo_texto=letras.getFontName();
                tamagno_letra=letras.getSize();
            }
            else if(menu=="12" | menu=="16" | menu=="20" | menu=="24") {
                estilo_letra=letras.getStyle();
                tipo_texto=letras.getFontName();
            }
            miarea.setFont(new Font(tipo_texto, estilo_letra,
tamagno_letra));
            if(estilo_letra==0) {
                tip="Normal";
            }else if(estilo_letra==1) {
                tip="Negrita";
            }else if(estilo_letra==2) {
                tip="Cursiva";
            }
            informa.setText("Tipo: " +tipo_texto +" Estilo: " + tip +
" Tamaño: " +tamagno_letra);
        }
    }
}
String tip;

```

```
JLabel informa;  
JTextPane miarea;  
JMenu fuente, estilo, tamagno;  
Font letras;  
}
```



Componentes Swing. Creación de procesador de textos. Práctica guiada VI. (Vídeo 106)

En el programa anterior había un conflicto cuando seleccionábamos negrita o cursiva cuando previamente habíamos seleccionado una.

Para controlar este error hemos modificados el siguiente código:

```
87     public void actionPerformed(ActionEvent e) {
88         // TODO Auto-generated method stub
89         letras=miarea.getFont();
90         if(menu=="Arial" || menu=="Courier" || menu=="Verdana") {
91             estilo_letra=letras.getStyle();
92             tamagno_letra=letras.getSize();
93         }
94         else if(menu=="Negrita" || menu=="Cursiva") {
95             if(letras.getStyle()==1 ||letras.getStyle()==2){
96                 estilo_letra=3;
97             }
98             tipo_texto=letras.getFontName();
99             tamagno_letra=letras.getSize();
100        }
101        else if(menu=="12" | menu=="16" | menu=="20" | menu=="24") {
102            estilo_letra=letras.getStyle();
103            tipo_texto=letras.getFontName();
104        }
105        miarea.setFont(new Font(tipo_texto, estilo_letra, tamagno_letra));
106        if(estilo_letra==0) {
107            tip="Normal";
108        }else if(estilo_letra==1) {
109            tip="Negrita";
110        }else if(estilo_letra==2) {
111            tip="Cursiva";
112        }
113        informa.setText("Tipo: " +tipo_texto + " Estilo: " + tip + " Tamaño: " +tamagno_letra);
114    }
115 }
```

```
else if(menu=="Negrita" || menu=="Cursiva") {
    if(letras.getStyle()==1 ||letras.getStyle()==2){
        estilo_letra=3;
    }
}
```

Con el valor 3 tenemos Negrita y Cursiva.

Una clase a comentar es StyledEditorKit nos permite manejar el texto que hay dentro de un componente Swing.

```
package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import java.awt.font.TextAttribute;

import javax.swing.*;
import javax.swing.text.StyledEditorKit;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10);
        configura_menu("Courier","fuente","Courier", 9,10);
        configura_menu("Verdana","fuente","Verdana", 9,10);

        configura_menu("Negrita","estilo","", Font.BOLD,1);
        configura_menu("Cursiva","estilo","", Font.ITALIC,1);

        configura_menu("12","tamaño","", 9,12);
        configura_menu("16","tamaño","", 9,16);
        configura_menu("20","tamaño","", 9,20);
        configura_menu("24","tamaño","", 9,24);

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
        add(miarea, BorderLayout.CENTER);
    }

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam) {
        JMenuItem elem_menu=new JMenuItem(rotulo);
        if(menu=="fuente") {
            fuente.add(elem_menu);
            if(tipo_letra=="Arial") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
            }
            else if(tipo_letra=="Courier") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
            }
            else if(tipo_letra=="Verdana") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
            }
        }
    }
}

```

```

    }
}
else if(menu=="estilo") {
    estilo.add(elem_menu);
    if(estilos==Font.BOLD) {
        elem_menu.addActionListener(new
StyledEditorKit.BoldAction());
    }else if(estilos==Font.ITALIC) {
        elem_menu.addActionListener(new
StyledEditorKit.ItalicAction());
    }
}
else if(menu=="tamaño") {
    tamagno.add(elem_menu);

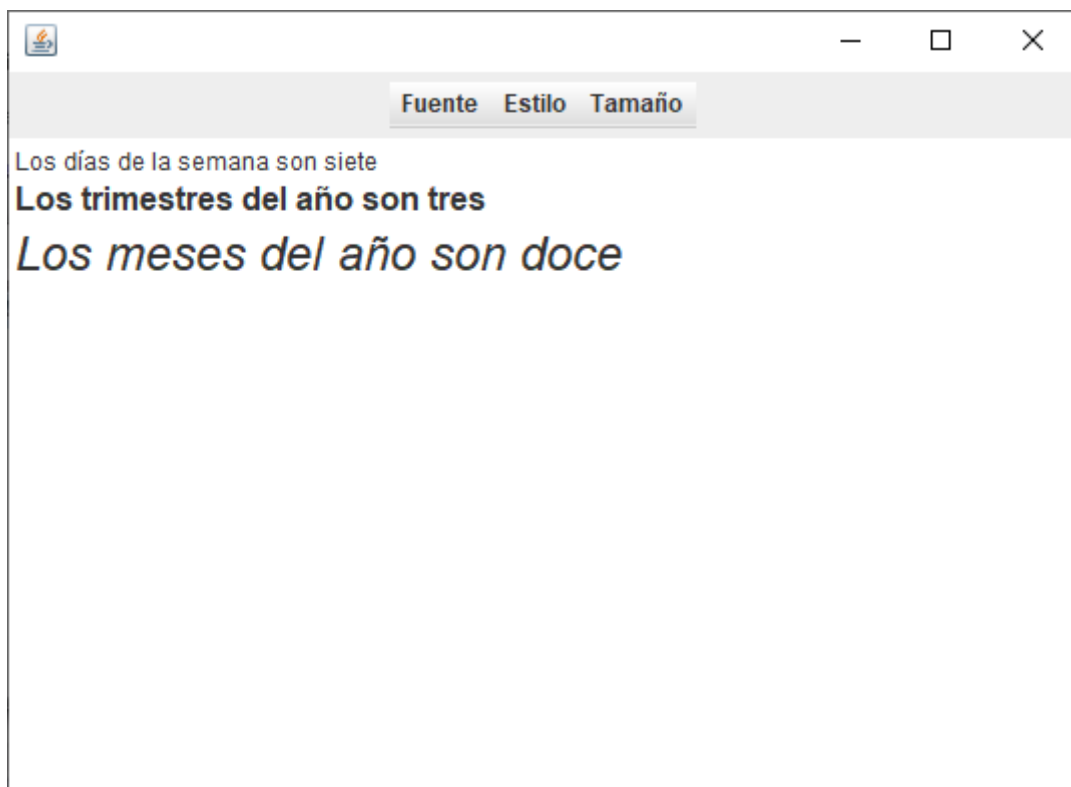
    elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
}

}

JTextPane miarea;
JMenu fuente, estilo, tamagno;
Font letras;
}

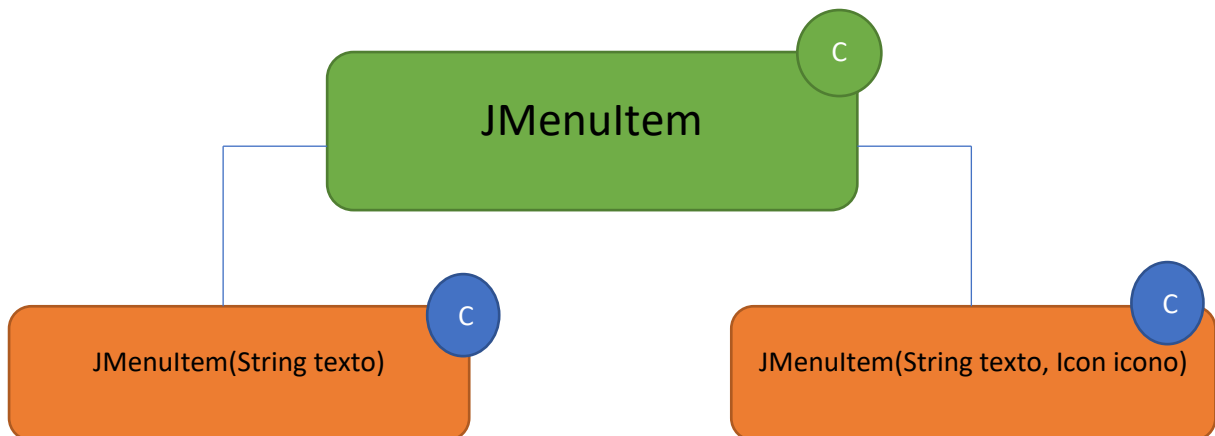
```

Este será el resultado:





Componentes Swing. Menús con imagen. (Vídeo 107)



Para este capítulo vamos a abrir un documento realizado con anterioridad llamado MarcoMenus.

```
package graficos;

import javax.swing.*;

public class MarcoMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuFrame mimarco=new MenuFrame();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuFrame extends JFrame{
    public MenuFrame() {
        setBounds(500,300,550,400);
        MenuLamina milamina=new MenuLamina();
        add(milamina);
        setVisible(true);
    }
}

class MenuLamina extends JPanel{
    public MenuLamina() {
        JMenuItem mibarra=new JMenuItem();
        JMenuItem archivo =new JMenuItem("Archivo");
        JMenuItem edicion =new JMenuItem("Edición");
        JMenuItem herramientas =new JMenuItem("Herramientas");

        JMenuItem guardar=new JMenuItem("Guardar");
        JMenuItem guardar_como=new JMenuItem("Guardar Como...");
```

```
        JMenuItem cortar=new JMenuItem("Cortar", new
        ImageIcon("src/graficos/cortar.png"));
        JMenuItem copiar=new JMenuItem("Copiar", new
        ImageIcon("src/graficos/copiar.png"));
        JMenuItem pegar=new JMenuItem("Pegar", new
        ImageIcon("src/graficos/pegar.png"));
```

Las imágenes las he guardado en la carpeta Gráficos, que a su vez está en la carpeta src.

```

JMenu opciones=new JMenu("Opciones");
JMenuItem opcion1=new JMenuItem("Opción 1");
JMenuItem opcion2=new JMenuItem("Opción 2");

JMenuItem generales=new JMenuItem("Generales");

archivo.add(guardar);
archivo.add(guardar_como);

edicion.add(cortar);
edicion.add(copiar);
edicion.add(pegar);
edicion.addSeparator();
edicion.add(opciones);

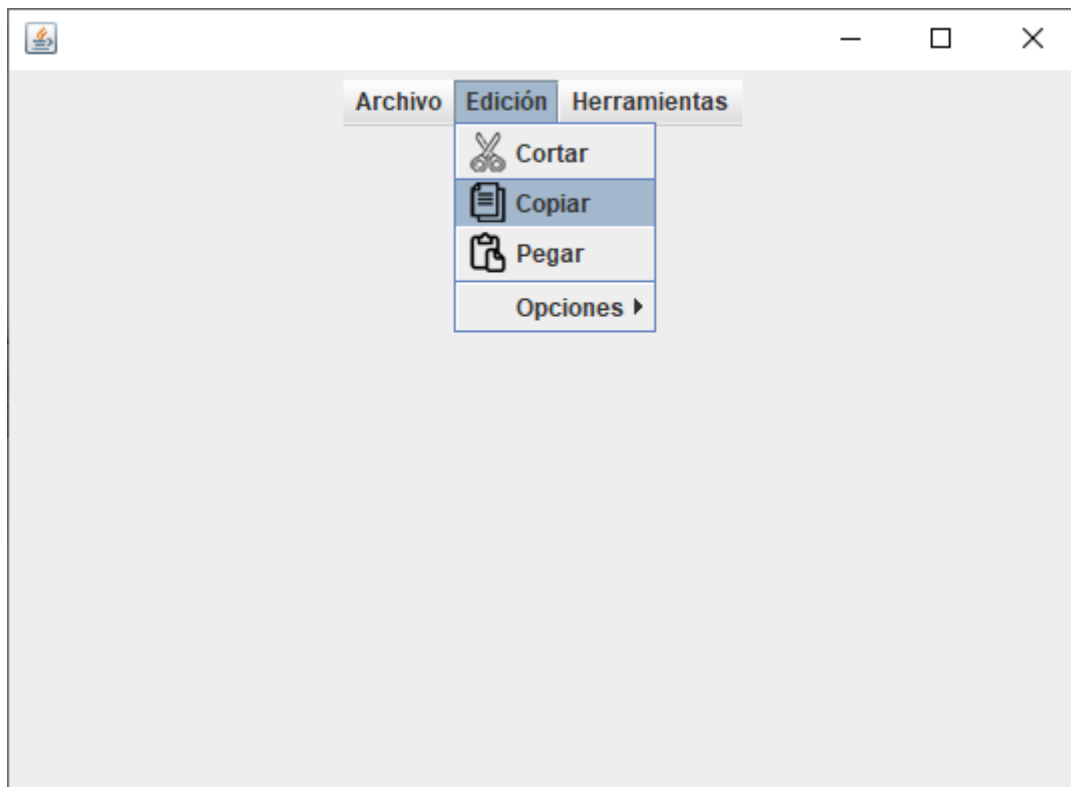
opciones.add(opcion1);
opciones.add(opcion2);

herramientas.add(generales);

mibarra.add(archivo);
mibarra.add(edicion);
mibarra.add(herramientas);
add(mibarra);
}
}

```

Este será el resultado:



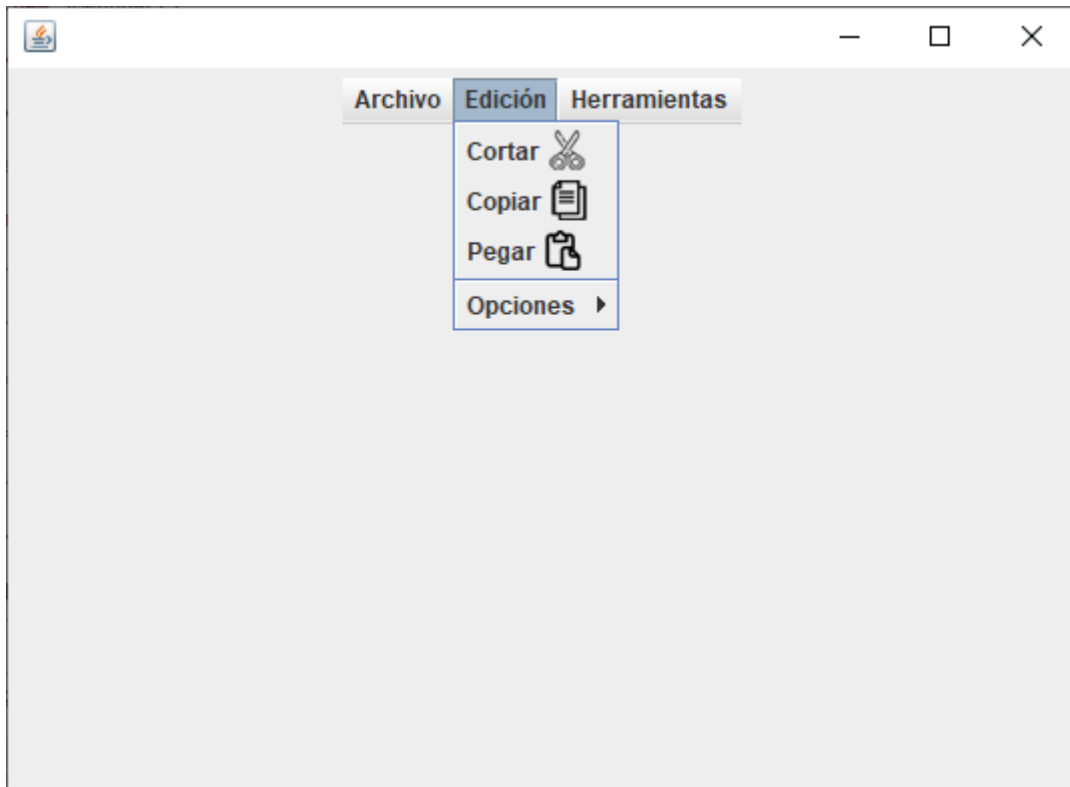
Si queremos que las imágenes aparezcan a la derecha agregaremos el siguiente código:


```

JMenuItem cortar=new JMenuItem("Cortar",new ImageIcon("src/graficos/cortar.png"));
cortar.setHorizontalTextPosition(SwingConstants.LEFT); ←
JMenuItem copiar=new JMenuItem("Copiar",new ImageIcon("src/graficos/copiar.png"));
copiar.setHorizontalTextPosition(SwingConstants.LEFT); ←
JMenuItem pegar=new JMenuItem("Pegar",new ImageIcon("src/graficos/pegar.png"));
pegar.setHorizontalTextPosition(SwingConstants.LEFT); ←
JMenu opciones=new JMenu("Opciones");
JMenuItem opcion1=new JMenuItem("Opción 1");
JMenuItem opcion2=new JMenuItem("Opción 2");

```

Este será el resultado:



Ahora en la aplicación del procesador de textos vamos a ver como agregar los iconos ya que en este proyecto utilizábamos un método para todas las opciones de menú.

```

package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import java.awt.font.TextAttribute;

import javax.swing.*;
import javax.swing.text.StyledEditorKit;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
class MenuProcesador_II extends JFrame{

```

```

public MenuProcesador_II() {
    setBounds(500,300,550,400);
    LaminaProcesador_II milamina=new LaminaProcesador_II();
    add(milamina);
    setVisible(true);
}
}

```

```

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

```

```

        configura_menu("Arial","fuente","Arial", 9,10,"");
        configura_menu("Courier","fuente","Courier", 9,10,"");
        configura_menu("Verdana","fuente","Verdana", 9,10,"");

        configura_menu("Negrita","estilo","",
Font.BOLD,1,"src/graficos/negrita.png");
        configura_menu("Cursiva","estilo","",
Font.ITALIC,1,"src/graficos/cursiva.png");

        configura_menu("12","tamaño","", 9,12,"");
        configura_menu("16","tamaño","", 9,16,"");
        configura_menu("20","tamaño","", 9,20,"");
        configura_menu("24","tamaño","", 9,24,"");

```

```

mibarra.add(fuente);
mibarra.add(estilo);
mibarra.add(tamagno);

```

```

laminamenu.add(mibarra);
add(laminamenu, BorderLayout.NORTH);

```

```

miarea=new JTextPane();
add(miarea, BorderLayout.CENTER);

```

```

}

```

```

public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam, String ruta_icono) {

```

```

    JMenuItem elem_menu=new JMenuItem(rotulo, new
ImageIcon(ruta_icono));
    if(menu=="fuente") {
        fuente.add(elem_menu);
        if(tipo_letra=="Arial") {
            elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
        }
        else if(tipo_letra=="Courier") {
            elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
        }
        else if(tipo_letra=="Verdana") {

```

Agregamos un parámetro más, los que no tienen icono irá "" y los que tienen icono irá la ruta.

Agregamos un nuevo parámetro de tipo ImageIcon donde irá la ruta de la imagen que tenemos guardada.

```

        elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
    }
    }
    else if(menu=="estilo") {
        estilo.add(elem_menu);
        if(estilos==Font.BOLD) {
            elem_menu.addActionListener(new
StyledEditorKit.BoldAction());
        }else if(estilos==Font.ITALIC) {
            elem_menu.addActionListener(new
StyledEditorKit.ItalicAction());
        }
    }
    else if(menu=="tamaño") {
        tamagno.add(elem_menu);

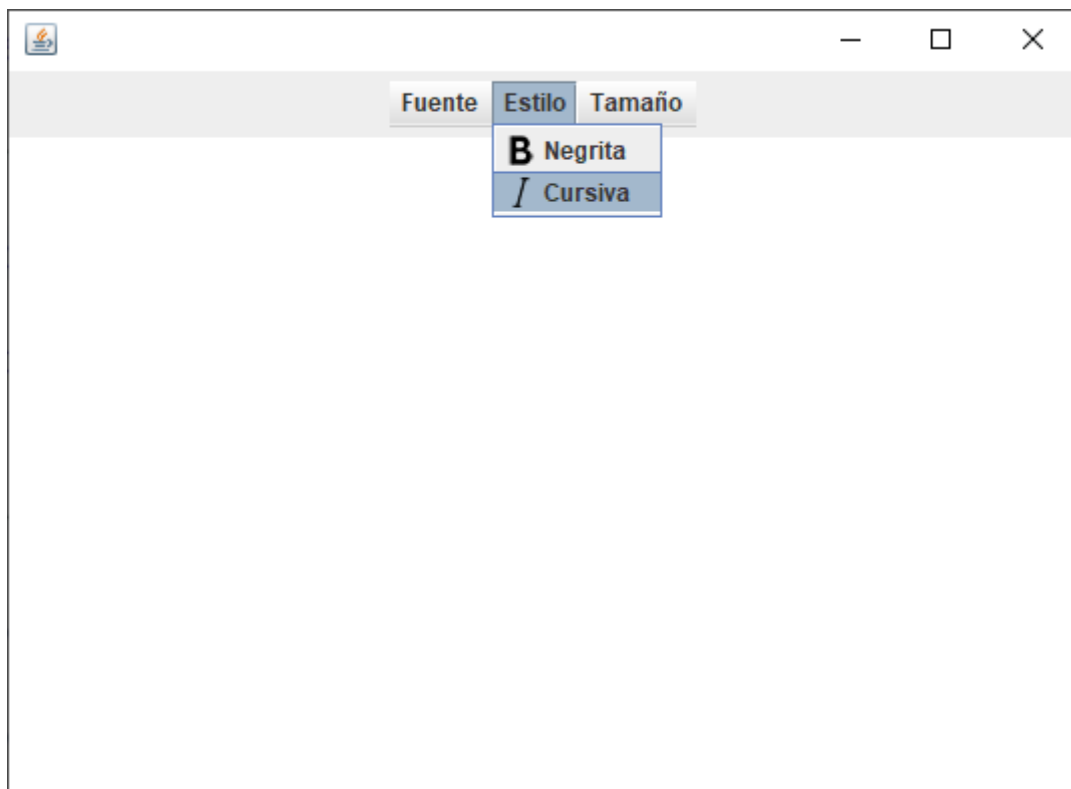
        elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
    }

}

JTextPane miarea;
JMenu fuente, estilo, tamagno;
Font letras;
}

```

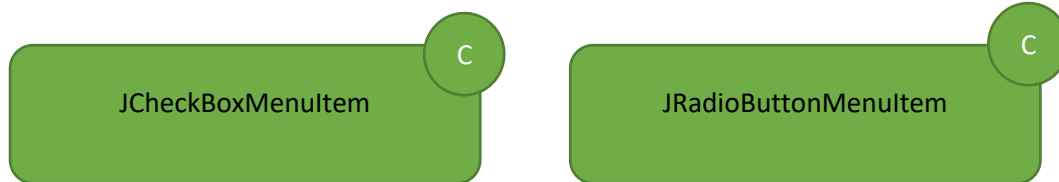
Este será el resultado:





Componentes Swing.Menús con CheckBox y RadioButtom. (Vídeo 108)

Menús con checkbox y radiobutton.



```
class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10,"");
        configura_menu("Courier","fuente","Courier", 9,10,"");
        configura_menu("Verdana","fuente","Verdana", 9,10,"");

        //configura_menu("Negrita","estilo","",
        Font.BOLD,1,"src/graficos/negrita.png");
        //configura_menu("Cursiva","estilo","",
        Font.ITALIC,1,"src/graficos/cursiva.png");

        JCheckBoxMenuItem negrita=new JCheckBoxMenuItem("Negrita", new
        ImageIcon("src/graficos/negrita.png"));
        JCheckBoxMenuItem cursiva=new JCheckBoxMenuItem("Cursiva", new
        ImageIcon("src/graficos/cursiva.png"));
        estilo.add(negrita);
        estilo.add(cursiva);
        negrita.addActionListener(new StyledEditorKit.BoldAction());
        cursiva.addActionListener(new StyledEditorKit.ItalicAction());

        configura_menu("12","tamaño","", 9,12,"");
        configura_menu("16","tamaño","", 9,16,"");
        configura_menu("20","tamaño","", 9,20,"");
        configura_menu("24","tamaño","", 9,24,"");

        mibarra.add(fuente);
        mibarra.add(estilo);
        mibarra.add(tamagno);

        laminamenu.add(mibarra);
        add(laminamenu, BorderLayout.NORTH);

        miarea=new JTextPane();
        add(miarea, BorderLayout.CENTER);
    }
}
```

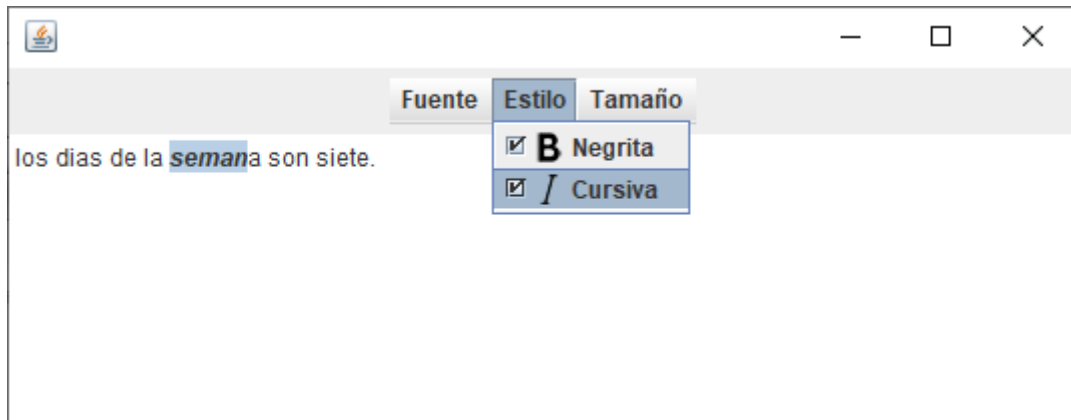
```

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam, String ruta_icono) {
        JMenuItem elem_menu=new JMenuItem(rotulo, new
ImageIcon(ruta_icono));
        if(menu=="fuente") {
            fuente.add(elem_menu);
            if(tipo_letra=="Arial") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
            }
            else if(tipo_letra=="Courier") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
            }
            else if(tipo_letra=="Verdana") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
            }
        }else if(menu=="tamaño") {
            tamagno.add(elem_menu);

            elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
        }
    }
}

```

Este será el resultado:



Ahora vamos con los tamaños de letra.

```

package graficos;

import java.awt.BorderLayout;

import java.awt.Font;

import java.awt.event.*;

import java.awt.font.TextAttribute;

import javax.swing.*;

import javax.swing.text.StyledEditorKit;

```

```

public class Procesador_II {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial","fuente","Arial", 9,10,"");
        configura_menu("Courier","fuente","Courier", 9,10,"");
        configura_menu("Verdana","fuente","Verdana", 9,10,"");
    }
}

```

```

        JCheckBoxMenuItem negrita=new JCheckBoxMenuItem("Negrita", new
        ImageIcon("src/graficos/negrita.png"));

        JCheckBoxMenuItem cursiva=new JCheckBoxMenuItem("Cursiva", new
        ImageIcon("src/graficos/cursiva.png"));

        estilo.add(negrita);

        estilo.add(cursiva);

        negrita.addActionListener(new StyledEditorKit.BoldAction());

        cursiva.addActionListener(new StyledEditorKit.ItalicAction());

```

```

        ButtonGroup tamagno_letra=new ButtonGroup();

        JRadioButtonMenuItem doce=new JRadioButtonMenuItem("12");

        JRadioButtonMenuItem dieciseis=new JRadioButtonMenuItem("16");

        JRadioButtonMenuItem veinte=new JRadioButtonMenuItem("20");

        JRadioButtonMenuItem veinticuatro=new JRadioButtonMenuItem("24");

        tamagno_letra.add(doce);

        tamagno_letra.add(dieciseis);

        tamagno_letra.add(veinte);

        tamagno_letra.add(veinticuatro);

        doce.addActionListener(new StyledEditorKit.FontSizeAction("cambia_tamaño", 12));

        dieciseis.addActionListener(new StyledEditorKit.FontSizeAction("cambia_tamaño",
16));

        veinte.addActionListener(new StyledEditorKit.FontSizeAction("cambia_tamaño", 20));

        veinticuatro.addActionListener(new StyledEditorKit.FontSizeAction("cambia_tamaño",
24));

        tamagno.add(doce);

        tamagno.add(dieciseis);

        tamagno.add(veinte);

        tamagno.add(veinticuatro);

```

```

        mibarra.add(fuente);

        mibarra.add(estilo);

        mibarra.add(tamagno);

```



```

laminamenu.add(mibarra);

add(laminamenu, BorderLayout.NORTH);

miarea=new JTextPane();
add(miarea, BorderLayout.CENTER);

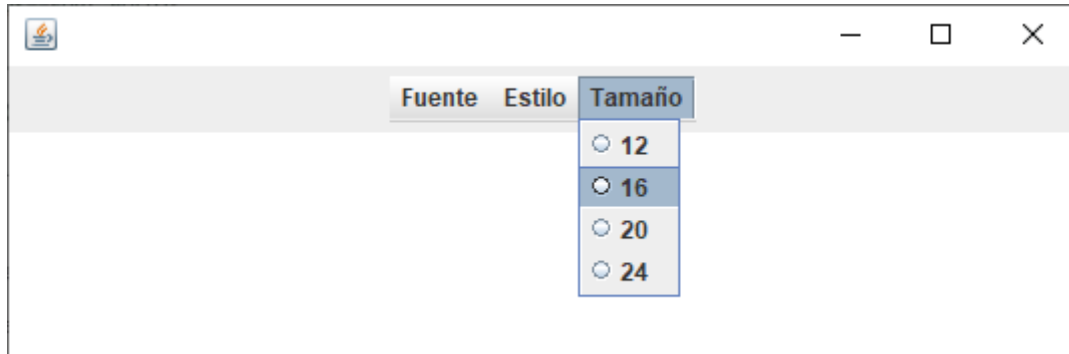
}

public void configura_menu(String rotulo, String menu, String tipo_letra, int estilos, int
tam, String ruta_icono) {
    JMenuItem elem_menu=new JMenuItem(rotulo, new ImageIcon(ruta_icono));
    if(menu=="fuente") {
        fuente.add(elem_menu);
        if(tipo_letra=="Arial") {
            elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
        }
        else if(tipo_letra=="Courier") {
            elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
        }
        else if(tipo_letra=="Verdana") {
            elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
        }
    }
    else if(menu=="tamaño") {
        tamagno.add(elem_menu);
        elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
    }
}
}

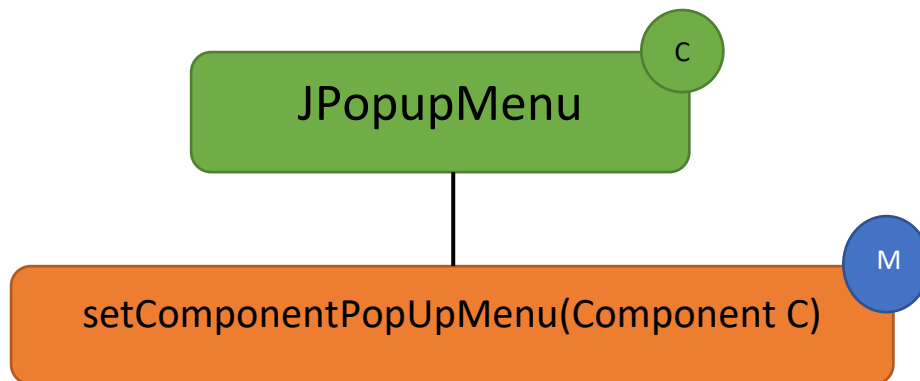
```

```
JTextPane miarea;  
JMenu fuente, estilo, tamagno;  
Font letras;  
}
```

Este será el resultado:

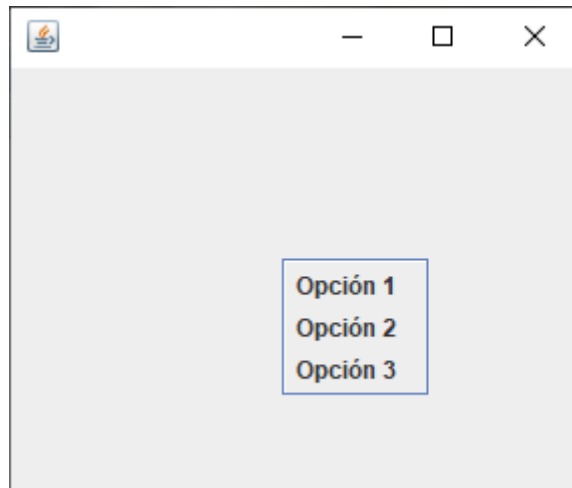


Componentes Swing. Menús emergentes. (Vídeo 109)



```
1 package graficos;
2
3 import javax.swing.*;
4
5 public class MarcoMEmergente {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         MarcoEmergenteM mimarco=new MarcoEmergenteM();
10        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11    }
12
13 }
14 class MarcoEmergenteM extends JFrame{
15     public MarcoEmergenteM() {
16         setBounds(100,100,300,250);
17         LaminaEmergenteM milamina=new LaminaEmergenteM();
18         add(milamina);
19         setVisible(true);
20     }
21 }
22
23 class LaminaEmergenteM extends JPanel{
24     public LaminaEmergenteM() {
25         JPopupMenu emergente=new JPopupMenu();
26         JMenuItem opcion1=new JMenuItem("Opción 1");
27         JMenuItem opcion2=new JMenuItem("Opción 2");
28         JMenuItem opcion3=new JMenuItem("Opción 3");
29         emergente.add(opcion1);
30         emergente.add(opcion2);
31         emergente.add(opcion3);
32
33         setComponentPopupMenu(emergente);
34     }
35 }
```

Este será el resultado cuando seleccionemos con el botón derecho del ratón.



Pero si ponemos componentes en la lamina y queremos que en un TextArea es donde queremos que salga el menú realizaremos el siguiente código:

```

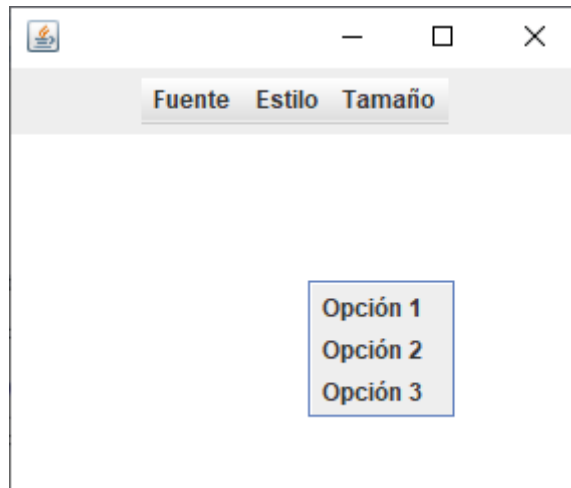
1 package graficos;
2 import java.awt.*;
3 import javax.swing.*;
4 public class MarcoEmergente {
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         MarcoEmergenteM mimarco=new MarcoEmergenteM();
8         mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9     }
10 }
11 class MarcoEmergenteM extends JFrame{
12     public MarcoEmergenteM() {
13         setBounds(100,100,300,250);
14         LaminaEmergenteM milamina=new LaminaEmergenteM();
15         add(milamina);
16         setVisible(true);
17     }
18 }
19 class LaminaEmergenteM extends JPanel{
20     public LaminaEmergenteM() {
21         setLayout(new BorderLayout());
22         JPanel laminamenu=new JPanel();
23         JMenuBar mibarra=new JMenuBar();
24         JMenu fuente =new JMenu("Fuente");
25         JMenu estilo =new JMenu("Estilo");
26         JMenu tamagno =new JMenu("Tamaño");
27         mibarra.add(fuente);
28         mibarra.add(estilo);
29         mibarra.add(tamagno);
30         laminamenu.add(mibarra);
31         add(laminamenu, BorderLayout.NORTH);
32         JTextPane miarea=new JTextPane();
33         add(miarea, BorderLayout.CENTER);
34         JPopupMenu emergente=new JPopupMenu();
35         JMenuItem opcion1=new JMenuItem("Opción 1");
36         JMenuItem opcion2=new JMenuItem("Opción 2");
37         JMenuItem opcion3=new JMenuItem("Opción 3");
38         emergente.add(opcion1);
39         emergente.add(opcion2);
40         emergente.add(opcion3);
41
  
```

```

43     miarea.setComponentPopupMenu(emergente);
44 }
45 }

```

Solo saldrá el menú emergente si hacemos botón derecho sobre el JTextPane.



Ahora con los conocimientos adquiridos vamos a llevarnos estos ejemplos con el proyecto del tratamiento de textos.

```

package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import java.awt.font.TextAttribute;

import javax.swing.*;
import javax.swing.text.StyledEditorKit;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();
        //-----
        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");

```

```

tamagno=new JMenu("Tamaño");

configura_menu("Arial", "fuente", "Arial", 9,10, "");
configura_menu("Courier", "fuente", "Courier", 9,10, "");
configura_menu("Verdana", "fuente", "Verdana", 9,10, "");

JCheckBoxMenuItem negrita=new JCheckBoxMenuItem("Negrita", new
ImageIcon("src/graficos/negrita.png"));
JCheckBoxMenuItem cursiva=new JCheckBoxMenuItem("Cursiva", new
ImageIcon("src/graficos/cursiva.png"));
estilo.add(negrita);
estilo.add(cursiva);
negrita.addActionListener(new StyledEditorKit.BoldAction());
cursiva.addActionListener(new StyledEditorKit.ItalicAction());

ButtonGroup tamagno_letra=new ButtonGroup();
JRadioButtonMenuItem doce=new JRadioButtonMenuItem("12");
JRadioButtonMenuItem dieciseis=new JRadioButtonMenuItem("16");
JRadioButtonMenuItem veinte=new JRadioButtonMenuItem("20");
JRadioButtonMenuItem veinticuatro=new JRadioButtonMenuItem("24");
tamagno_letra.add(doce);
tamagno_letra.add(dieciseis);
tamagno_letra.add(veinte);
tamagno_letra.add(veinticuatro);
doce.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 12));
dieciseis.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 16));
veinte.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 20));
veinticuatro.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 24));
tamagno.add(doce);
tamagno.add(dieciseis);
tamagno.add(veinte);
tamagno.add(veinticuatro);

mibarra.add(fuente);
mibarra.add(estilo);
mibarra.add(tamagno);

laminamenu.add(mibarra);
add(laminamenu, BorderLayout.NORTH);

miarea=new JTextPane();
add(miarea, BorderLayout.CENTER);

```

```

JPopupMenu emergente=new JPopupMenu();
JMenuItem negritaE=new JMenuItem("Negrita");
JMenuItem cursivaE=new JMenuItem("Cursiva");

emergente.add(negritaE);
emergente.add(cursivaE);

miarea.setComponentPopupMenu(emergente);

negritaE.addActionListener(new StyledEditorKit.BoldAction());
cursivaE.addActionListener(new StyledEditorKit.ItalicAction());

```

```

    }

    public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam, String ruta_icono) {
        JMenuItem elem_menu=new JMenuItem(rotulo, new
ImageIcon(ruta_icono));
        if(menu=="fuente") {
            fuente.add(elem_menu);
            if(tipo_letra=="Arial") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
            }
            else if(tipo_letra=="Courier") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
            }
            else if(tipo_letra=="Verdana") {
                elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
            }
        }

        else if(menu=="tamaño") {
            tamagno.add(elem_menu);

            elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
        }

    }

    JTextPane miarea;
    JMenu fuente, estilo, tamagno;
    Font letras;
}

```

Agregamos el código que está en el recuadro, este será el resultado cuando seleccionemos un texto y le demos con el botón derecho del ratón.



Ahora podemos acceder desde el menú y con el botón derecho del ratón.

El problema es que en el menú no muestra si se ha activado la negrita o la cursiva, esto puede generar confusión vamos a quitar los checkbox y lo dejamos como estaba antes el proyecto.

```
package graficos;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.*;
import java.awt.font.TextAttribute;

import javax.swing.*;
import javax.swing.text.StyledEditorKit;

public class Procesador_II {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MenuProcesador_II mimarco=new MenuProcesador_II();
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class MenuProcesador_II extends JFrame{
    public MenuProcesador_II() {
        setBounds(500,300,550,400);
        LaminaProcesador_II milamina=new LaminaProcesador_II();
        add(milamina);
        setVisible(true);
    }
}

class LaminaProcesador_II extends JPanel{
    public LaminaProcesador_II() {
        setLayout(new BorderLayout());
        JPanel laminamenu=new JPanel();
        JMenuBar mibarra=new JMenuBar();

        fuente=new JMenu("Fuente");
        estilo=new JMenu("Estilo");
        tamagno=new JMenu("Tamaño");

        configura_menu("Arial", "fuente", "Arial", 9,10,"");
        configura_menu("Courier", "fuente", "Courier", 9,10,"");
        configura_menu("Verdana", "fuente", "Verdana", 9,10,"");

        configura_menu("Negrita", "estilo", "", Font.BOLD,1, "src/graficos/negrita.png");
        configura_menu("Cursiva", "estilo", "", Font.ITALIC,1, "src/graficos/cursiva.png");

        ButtonGroup tamagno_letra=new ButtonGroup();
        JRadioButtonMenuItem doce=new JRadioButtonMenuItem("12");
        JRadioButtonMenuItem dieciseis=new JRadioButtonMenuItem("16");
        JRadioButtonMenuItem veinte=new JRadioButtonMenuItem("20");
        JRadioButtonMenuItem veinticuatro=new JRadioButtonMenuItem("24");
        tamagno_letra.add(doce);
        tamagno_letra.add(dieciseis);
        tamagno_letra.add(veinte);
        tamagno_letra.add(veinticuatro);
    }
}
```



```

doce.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 12));
dieciseis.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 16));
veinte.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 20));
veinticuatro.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", 24));
tamagno.add(doce);
tamagno.add(dieciseis);
tamagno.add(veinte);
tamagno.add(veinticuatro);

mibarra.add(fuente);
mibarra.add(estilo);
mibarra.add(tamagno);

laminamenu.add(mibarra);
add(laminamenu, BorderLayout.NORTH);

miarea=new JTextPane();
add(miarea, BorderLayout.CENTER);

JPopupMenu emergente=new JPopupMenu();
JMenuItem negritaE=new JMenuItem("Negrita");
JMenuItem cursivaE=new JMenuItem("Cursiva");

emergente.add(negritaE);
emergente.add(cursivaE);

miarea.setComponentPopupMenu(emergente);

negritaE.addActionListener(new StyledEditorKit.BoldAction());
cursivaE.addActionListener(new StyledEditorKit.ItalicAction());

}

public void configura_menu(String rotulo, String menu, String
tipo_letra, int estilos, int tam, String ruta_icono) {
JMenuItem elem_menu=new JMenuItem(rotulo, new
ImageIcon(ruta_icono));
if(menu=="fuente") {
fuente.add(elem_menu);
if(tipo_letra=="Arial") {
elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
}
else if(tipo_letra=="Courier") {
elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
}
else if(tipo_letra=="Verdana") {
elem_menu.addActionListener(new
StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
}
}else if(menu=="estilo"){
estilo.add(elem_menu);
if(estilos==Font.BOLD) {

```

```

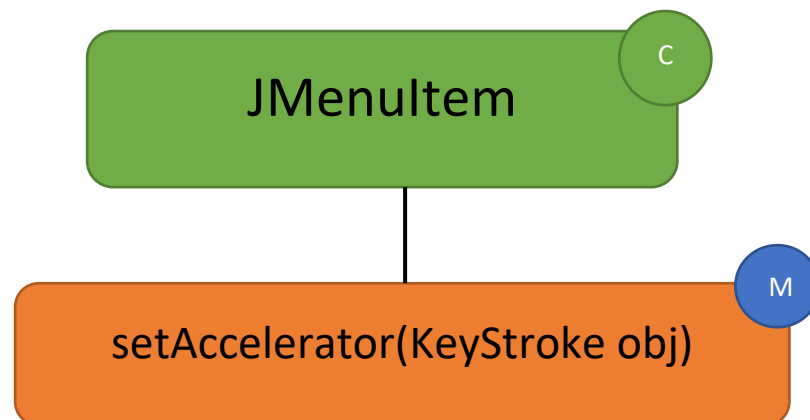
        elem_menu.addActionListener(new
StyledEditorKit.BoldAction());
    }else if(estilos==Font.ITALIC) {
        elem_menu.addActionListener(new
StyledEditorKit.ItalicAction());
    }
    }
    else if(menu=="tamaño") {
        tamagno.add(elem_menu);

        elem_menu.addActionListener(new
StyledEditorKit.FontSizeAction("cambia_tamaño", tam));
    }
}
JTextPane miarea;
JMenu fuente, estilo, tamagno;
Font letras;
}

```



Componentes Swing. Atajos del teclado. (Vídeo 110)



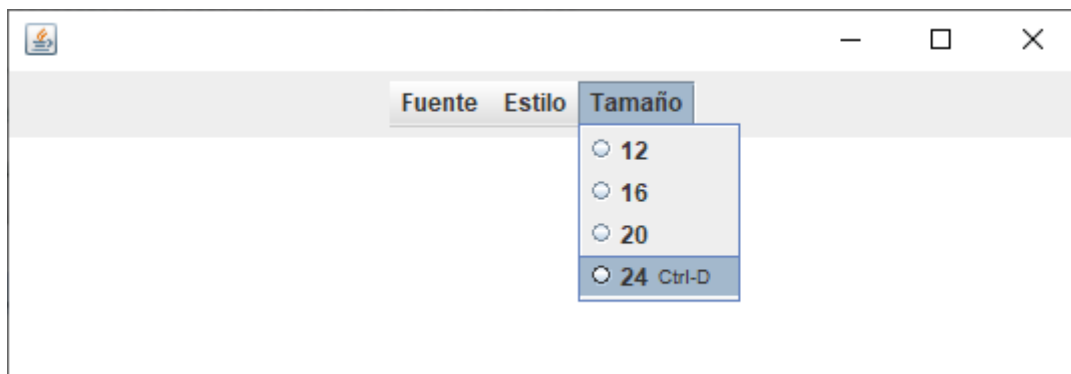
```
ButtonGroup tamagno_letra=new ButtonGroup();
JRadioButtonMenuItem doce=new JRadioButtonMenuItem("12");
JRadioButtonMenuItem dieciseis=new JRadioButtonMenuItem("16");
JRadioButtonMenuItem veinte=new JRadioButtonMenuItem("20");
JRadioButtonMenuItem veinticuatro=new JRadioButtonMenuItem("24");

veinticuatro.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D,InputEvent.CTRL_DOWN_MASK));

tamagno_letra.add(doce);
tamagno_letra.add(dieciseis);
tamagno_letra.add(veinte);
tamagno_letra.add(veinticuatro);
```

Queremos que del menú Tamaño la fuente 24 también funciones con Ctrl + D. tenemos que agregar el siguiente remarcado en el cuadro.

Al ejecutar la aplicación observaremos el siguiente detalle:



En el menú ya tenemos las teclas rápidas.

Ahora lo vamos a anular y lo vamos a aplicar para la negrita y la cursiva.

```
88 public void configura_menu(String rotulo, String menu, String tipo_letra, int estilos, int tam, String ru
89 JMenuItem elem_menu=new JMenuItem(rotulo, new ImageIcon(ruta_icono));
90 if(menu=="fuente") {
91     fuente.add(elem_menu);
92     if(tipo_letra=="Arial") {
93         elem_menu.addActionListener(new StyledEditorKit.FontFamilyAction("cambia_letra", "Arial"));
94     }
95     else if(tipo_letra=="Courier") {
96         elem_menu.addActionListener(new StyledEditorKit.FontFamilyAction("cambia_letra", "Courier"));
97     }
98 }
```

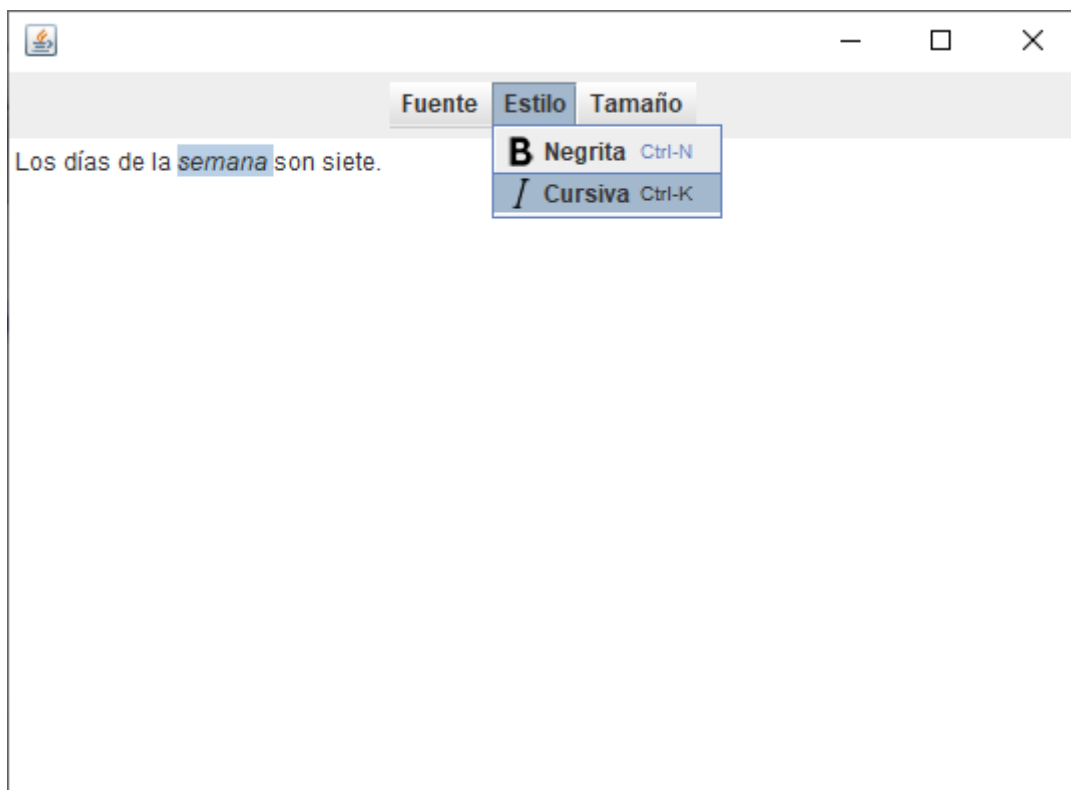
```

98     else if(tipo_letra=="Verdana") {
99         elem_menu.addActionListener(new StyledEditorKit.FontFamilyAction("cambia_letra", "Verdana"));
100     }
101 }else if(menu=="estilo"){
102     estilo.add(elem_menu);
103     if(estilos==Font.BOLD) {
104
105         elem_menu.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N, InputEvent.CTRL_DOWN_MASK));
106
107         elem_menu.addActionListener(new StyledEditorKit.BoldAction());
108
109     }else if(estilos==Font.ITALIC) {
110
111         elem_menu.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_K, InputEvent.CTRL_DOWN_MASK));
112
113         elem_menu.addActionListener(new StyledEditorKit.ItalicAction());
114
115     }

```

```
elem_menu.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N, InputEvent.CTRL_DOWN_MASK));
```

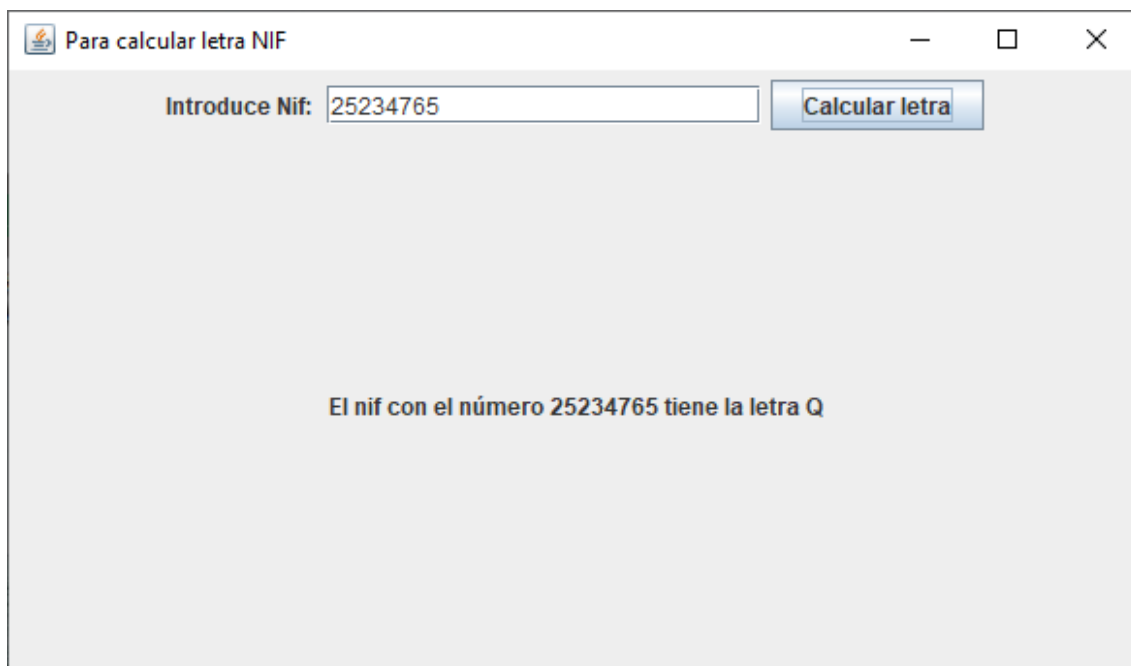
```
elem_menu.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_K, InputEvent.CTRL_DOWN_MASK));
```





Contenido

Layouts I (Vídeo 81).....	1
Layouts II. (Vídeo 82).....	5
Layouts III. (Vídeo 83).....	9
Layouts IV (Vídeo 84)	15
Layouts V. (Vídeo 85)	19
Componentes Swing. Cuadros de texto I. (Vídeo 86)	26
Componentes Swing. Cuadros de texto II. (Vídeo 87)	32



Componentes Swing. Eventos de cuadros de texto. (Vídeo 88)	37
Componentes Swing. Eventos de cuadros de texto II. (Vídeo 89)	41
Componentes Swing. Área de texto I. (Vídeo 90)	45
Componentes Swing. Áreas de texto II. (Vídeo 91).....	48
Componentes Swing CheckBox. (Vídeo 92)	52
Componentes Swing. Botones de radio. (Vídeo 93)	58
Componentes Swing. Botones de radio II. (Vídeo 94).....	60
Componentes Swing. ComboBox. (Vídeo 95)	64
Componentes Swing. JSlider I. (Vídeo 96).....	67
Componentes Swing. JSlider II. (Vídeo 97).....	69
Componentes Swing. JSpinner I. (Vídeo 98)	76
Componentes Swing. Jspinner II. (Vídeo 99).....	80
Componentes Swing. Creación de menús I. (Vídeo 100)	83
Componentes Swing. Creación de procesador de textos. Práctica guiada I. (Vídeo 101)	88
Java Componentes Swing. Creación de procesador de textos. Práctica guiada II. (Vídeo 102)..	91

Componentes Swing. Creación de procesador de textos. Práctica guiada III. (Vídeo 103)	95
Componentes Swing. Creación procesador de textos. Práctica guiada IV. (Vídeo 104)	97
Componentes Swing. Creación de procesador de textos. Práctica guiada V. (Vídeo 105)	100
Componentes Swing. Creación de procesador de textos. Práctica guiada VI. (Vídeo 106)	106
Componentes Swing. Menús con imagen. (Vídeo 107)	110
Componentes Swing. Menús con CheckBox y RadioButtom. (Vídeo 108).....	116
Componentes Swing. Menús emergentes. (Vídeo 109).....	122
Componentes Swing. Atajos del teclado. (Vídeo 110)	130