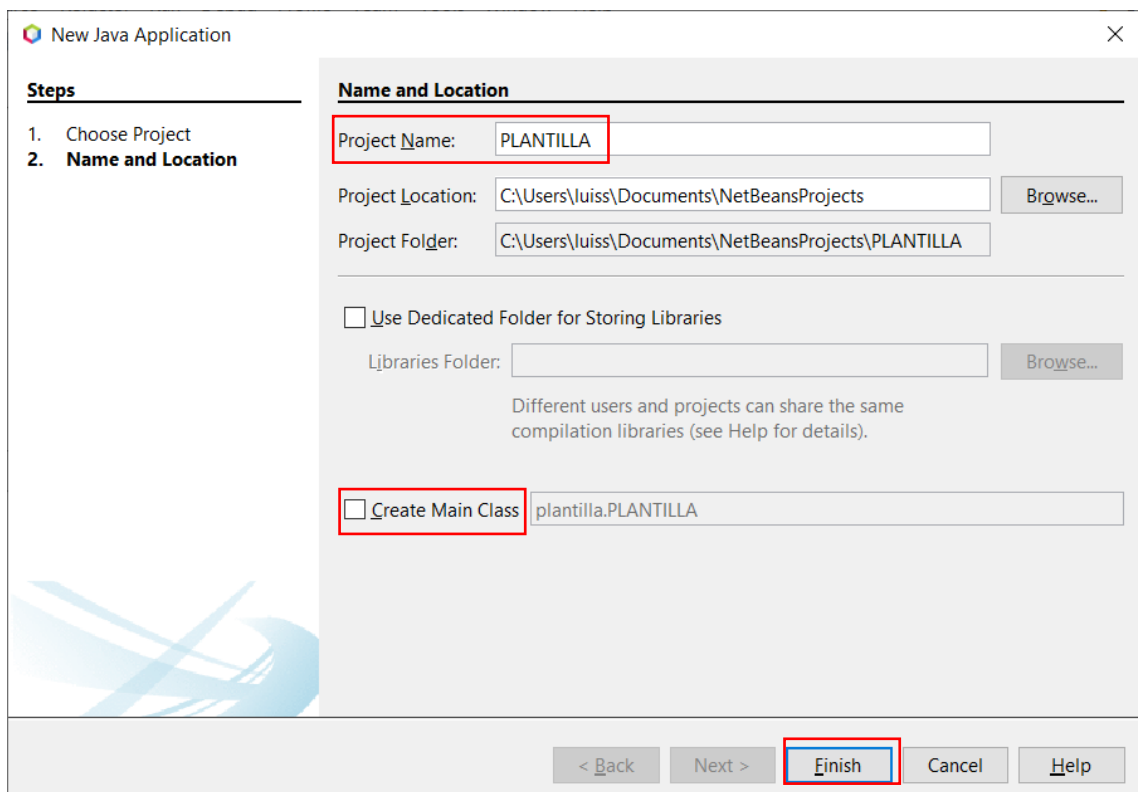
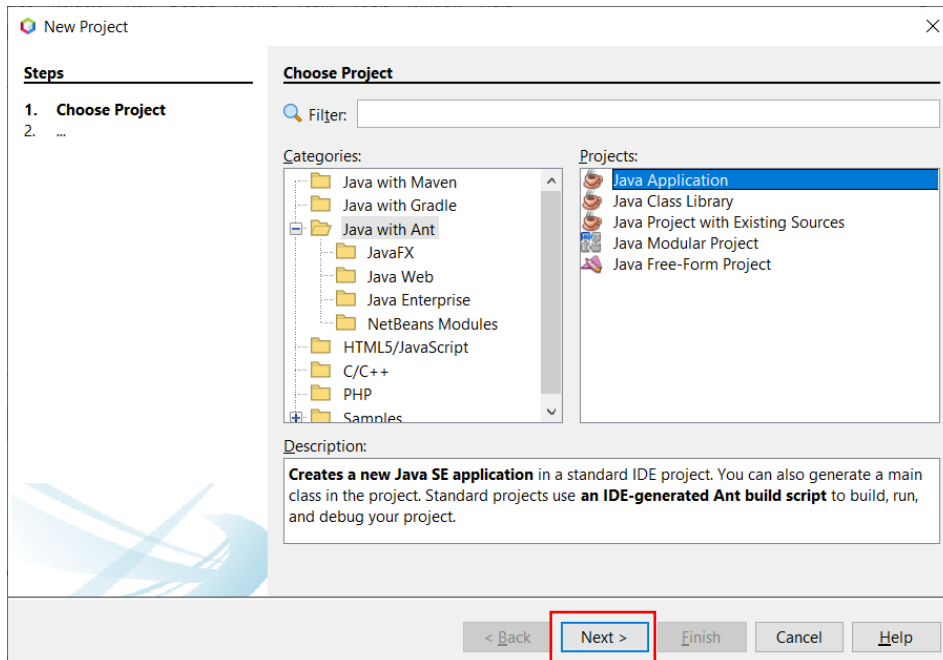
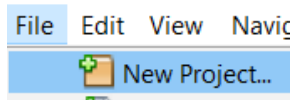
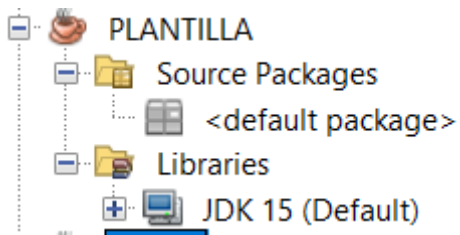
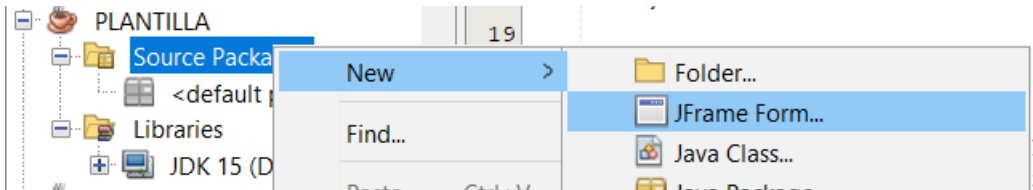


CREACION DE UN PROYECTO FORMULARIO (PLANTILLA)





CREACION DE UN FORMULARIO (PLANTILLA)



New JFrame Form

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: Vista

Project: PLANTILLA

Location: Source Packages

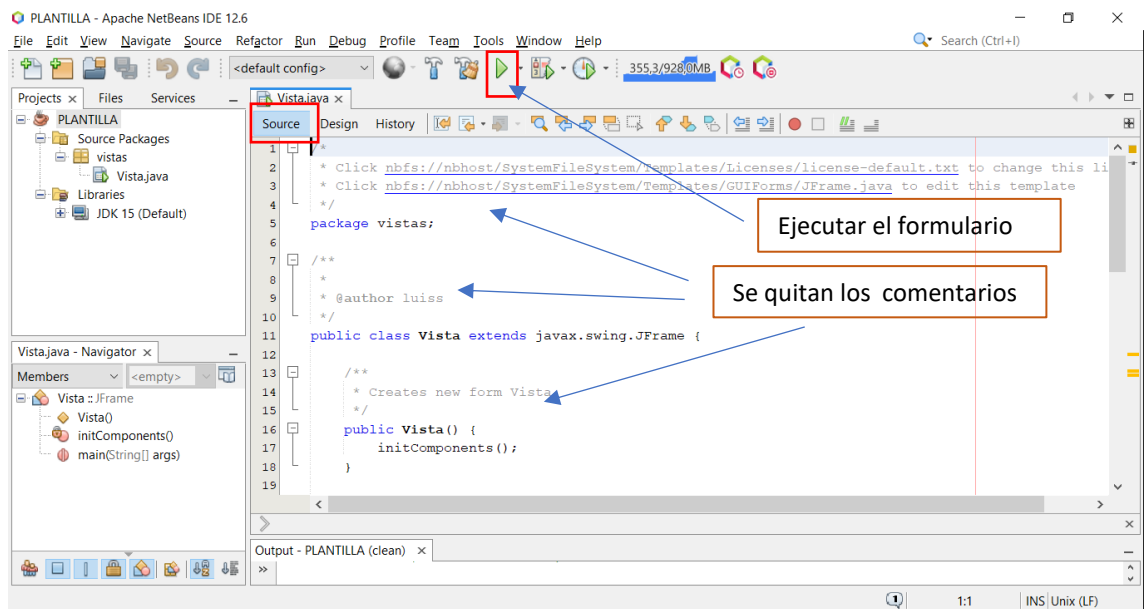
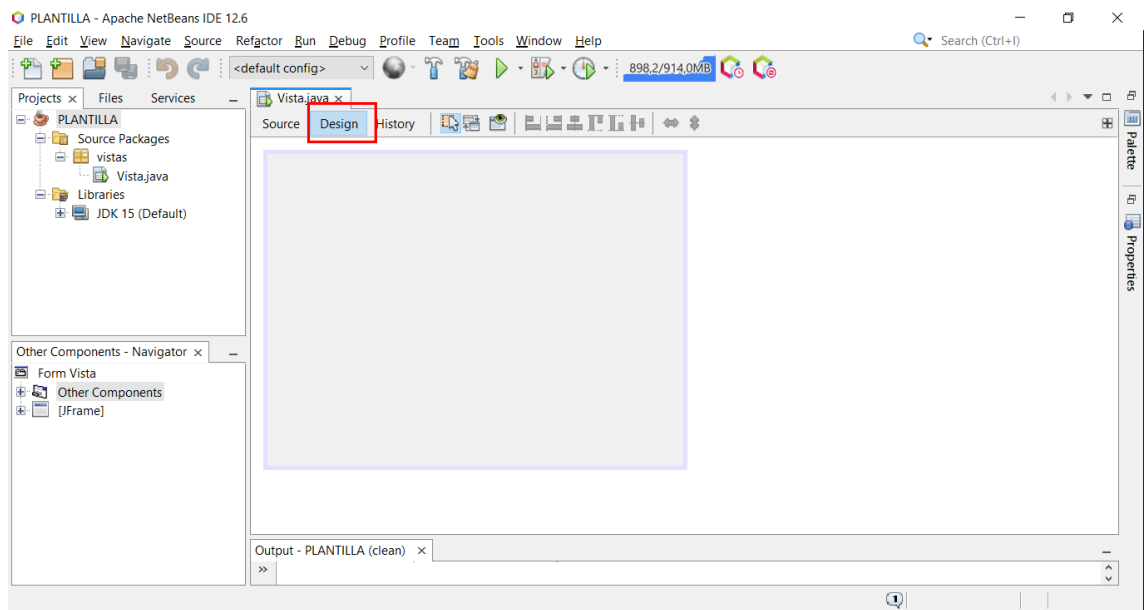
Package: vistas

Created File: C:\Users\luiss\Documents\NetBeansProjects\PLANTILLA\src\vistas\Vista.java

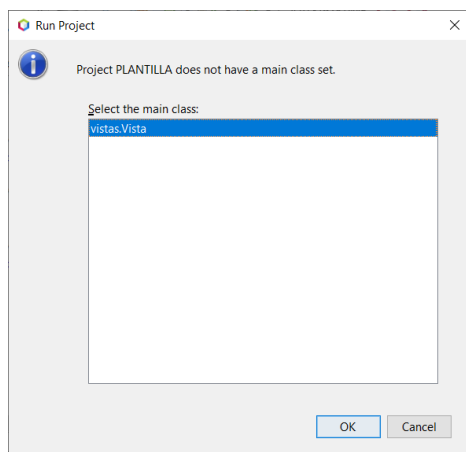
Superclass: Browse...

Interfaces: Browse...

< Back Next > **Finish** Cancel Help



Seleccionar la clase principal



Ejecución del formulario

Decoración del formulario

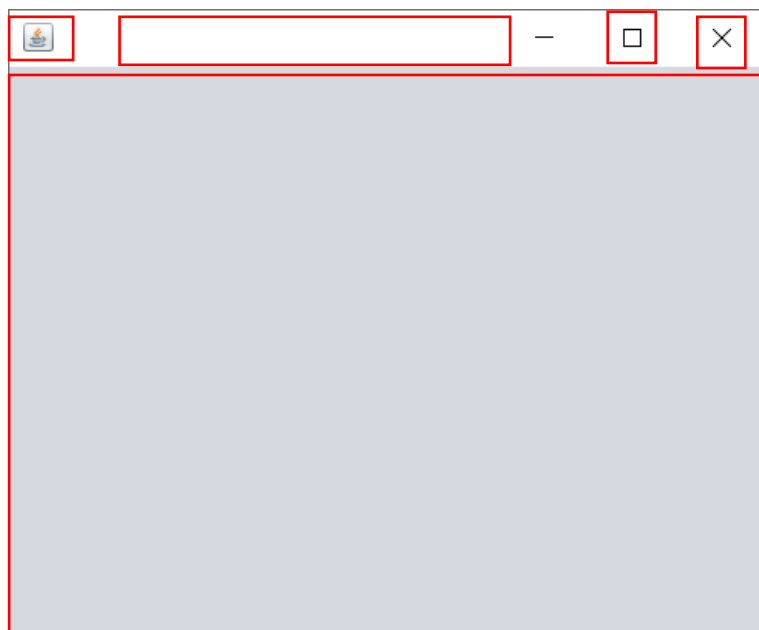
Icono del formulario

Título del formulario

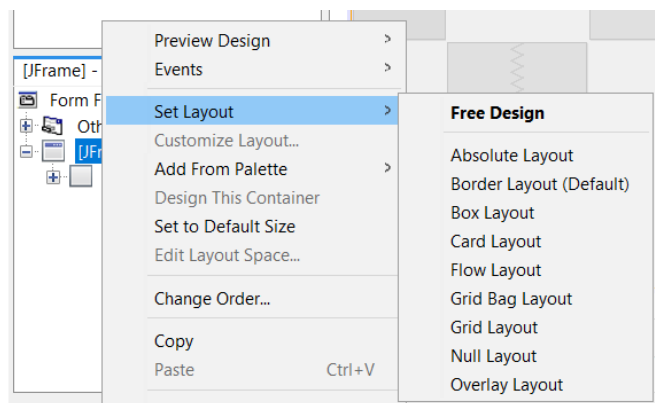
Redimensionar el formulario

Cerrar el formulario

Tamaño del formulario

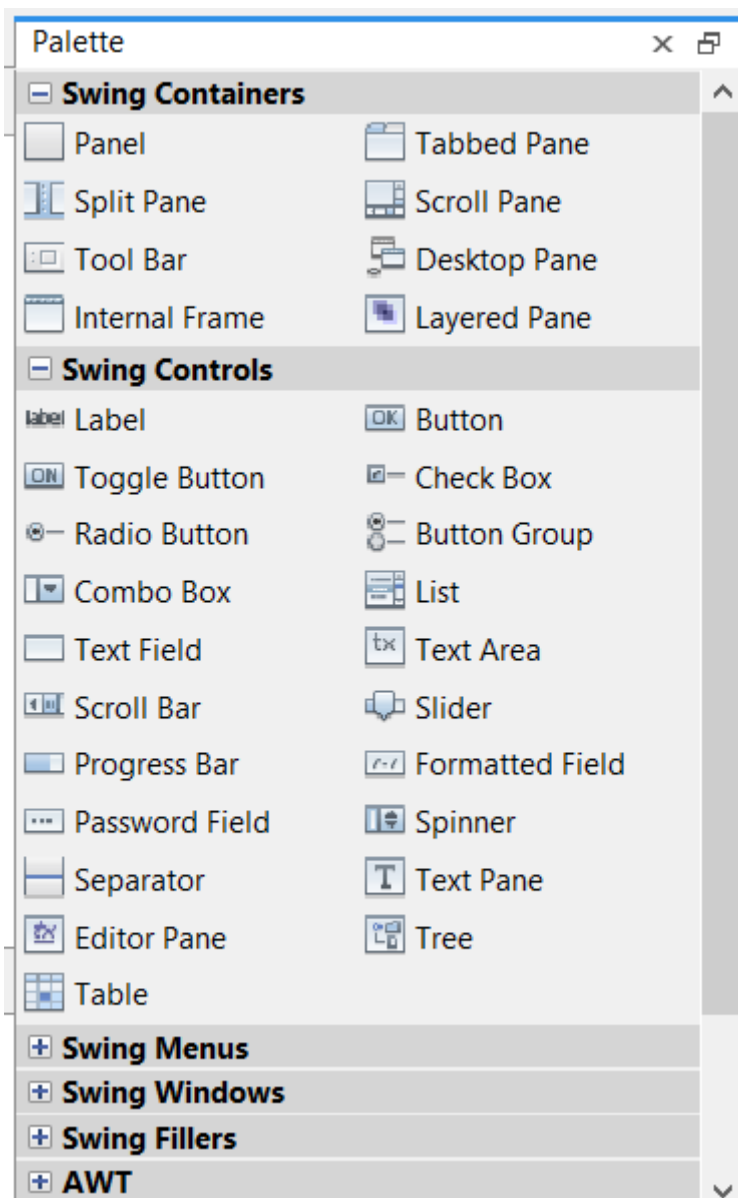


TIPOS DE LAYOUT



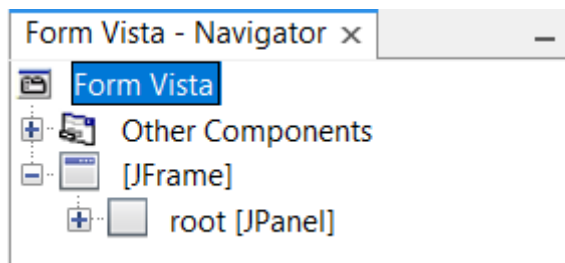
Layout	
Horizontal Size	Default
Vertical Size	Default
Horizontal Resizable	<input type="checkbox"/>
Vertical Resizable	<input type="checkbox"/>

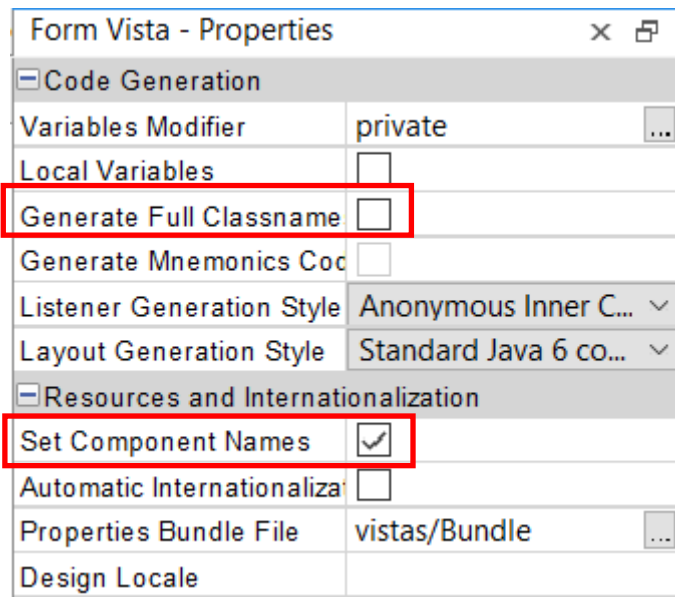
PALETA DE CONTROLES



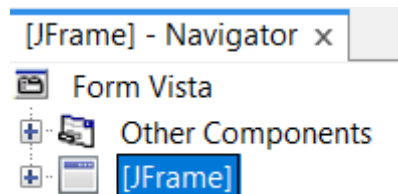
PROPIEDADES

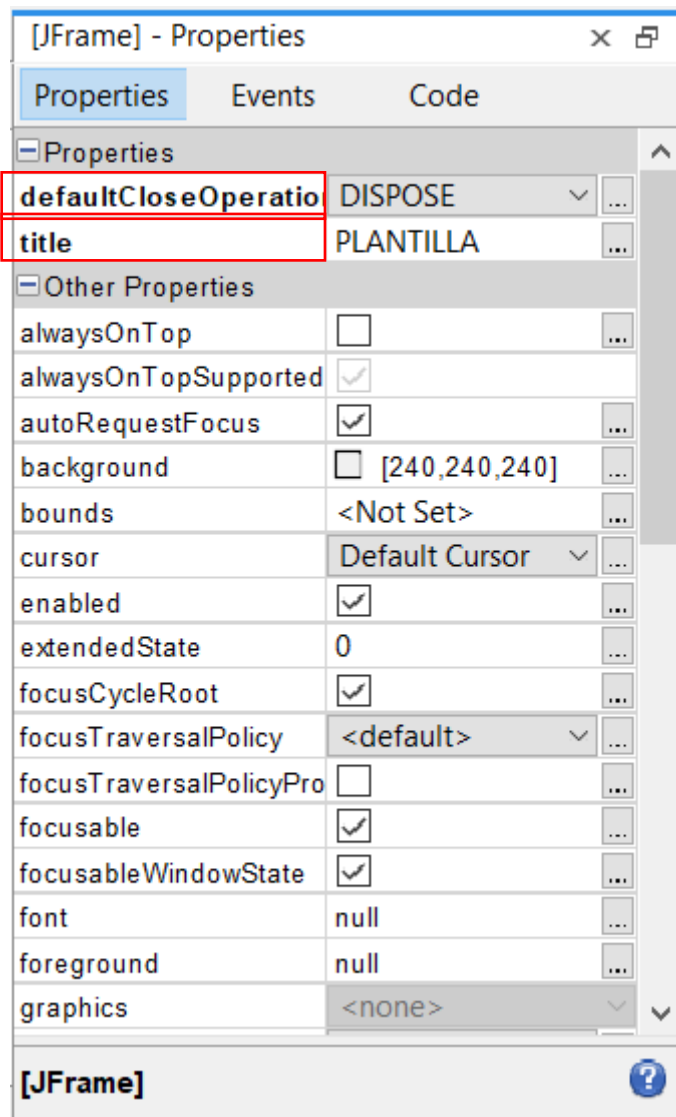
Ejemplo de propiedades y código de Form

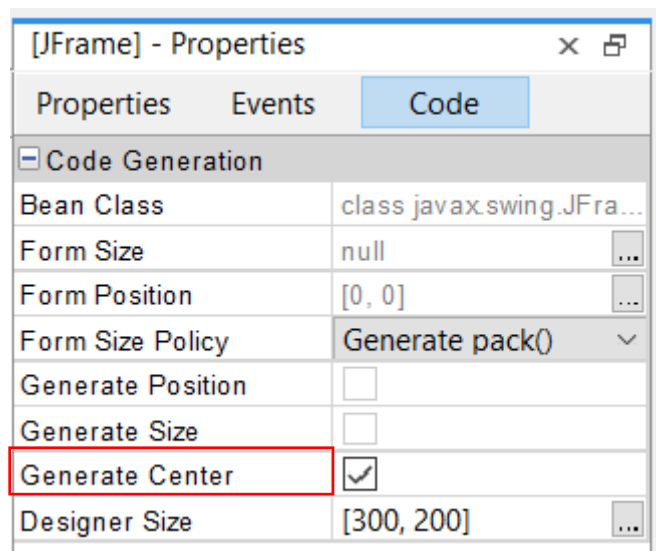
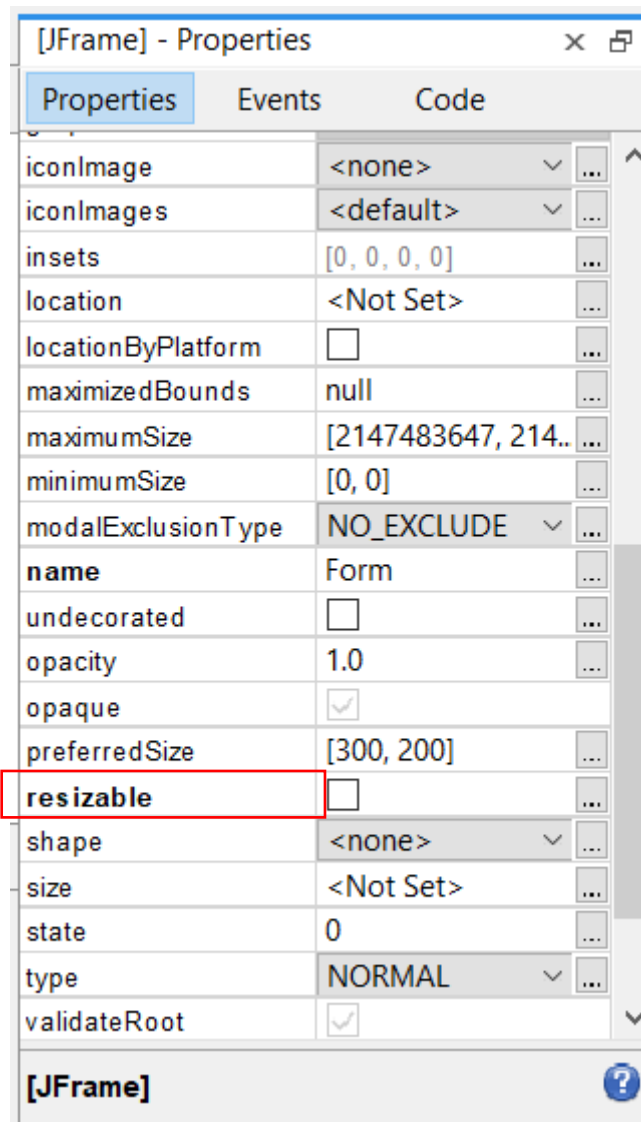




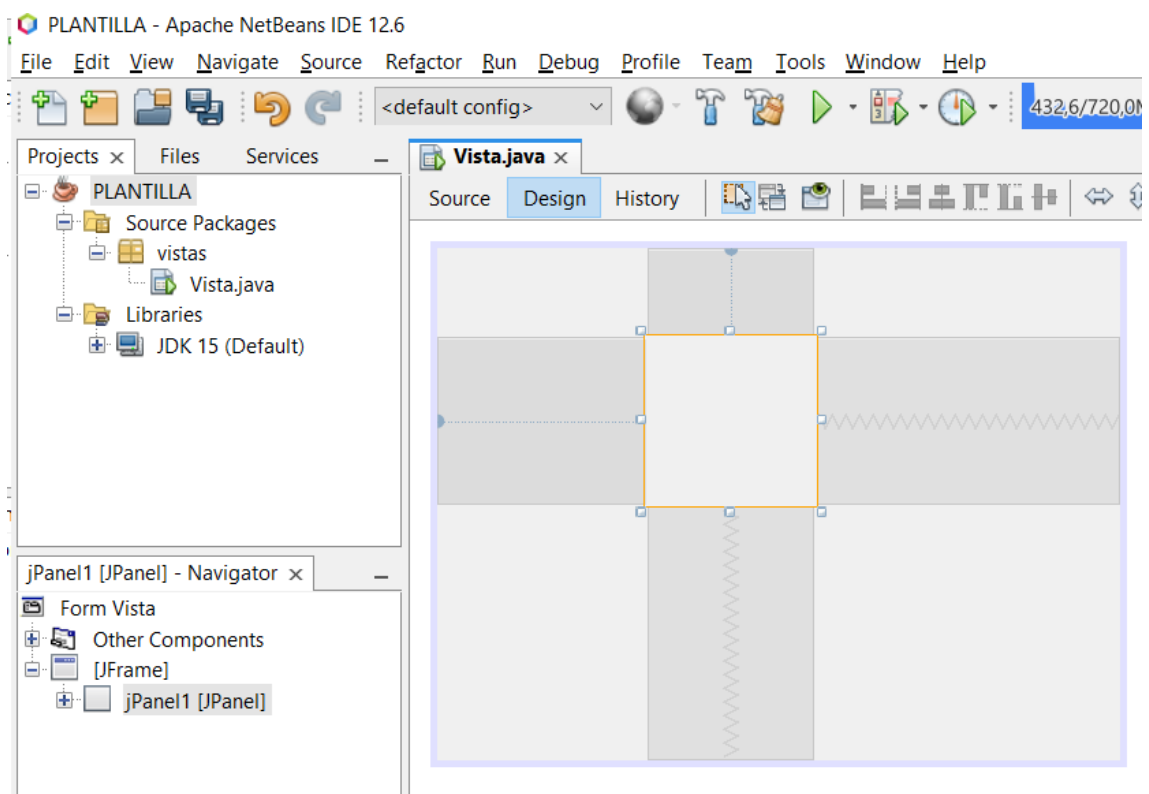
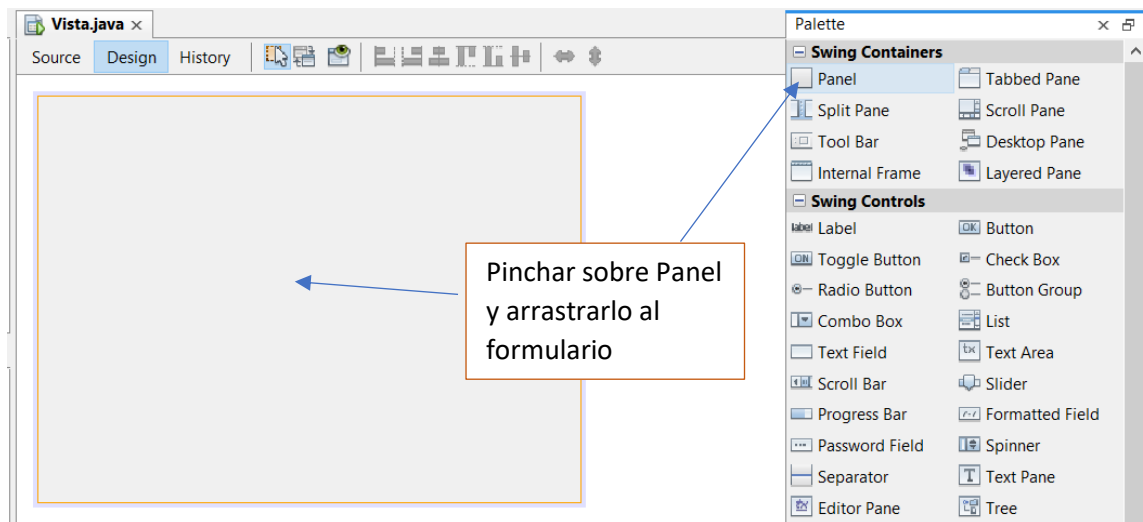
Ejemplo de propiedades y código del JFrame



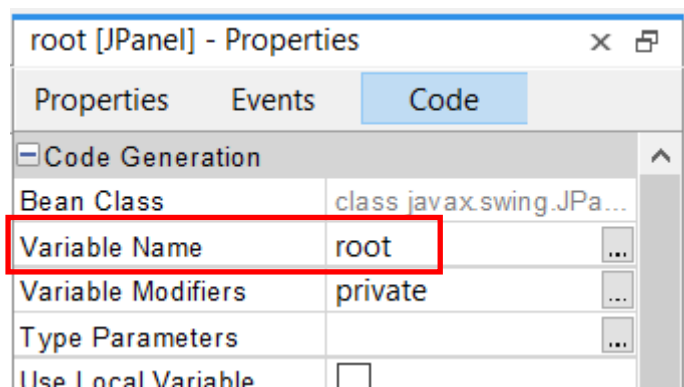


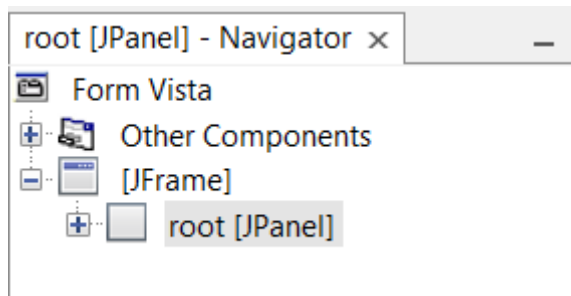


Ejemplo de propiedades de un control JPanel



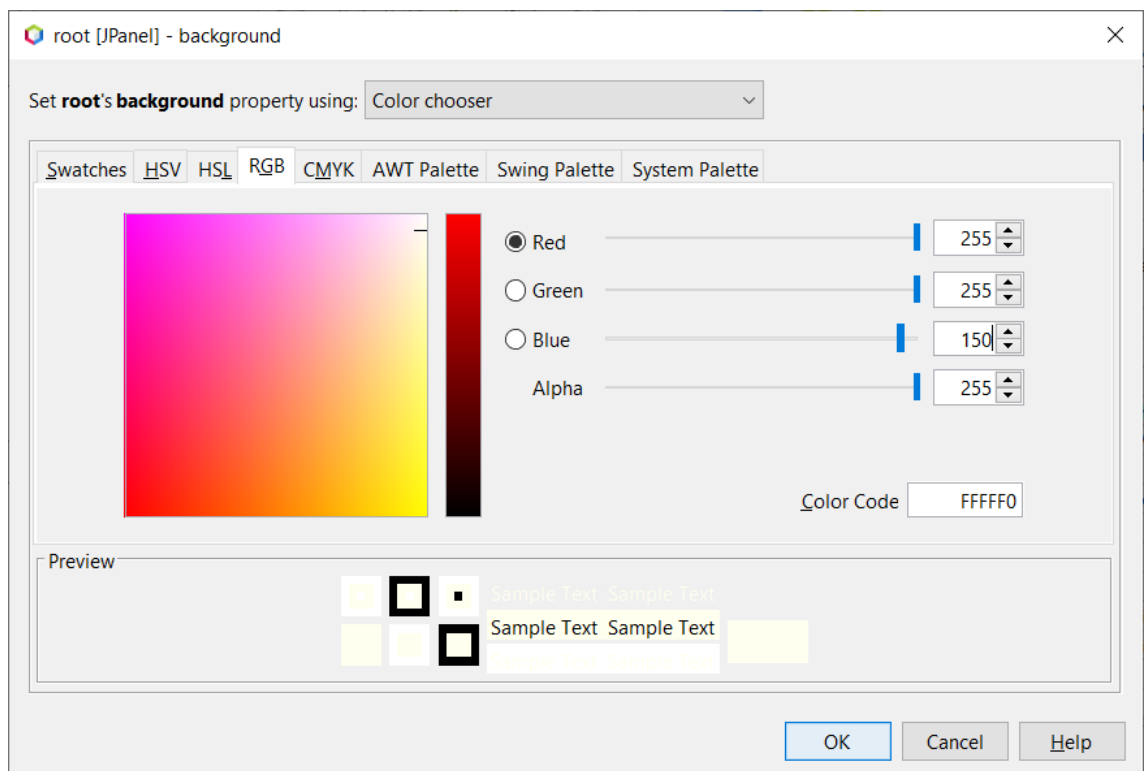
Cambiar el nombre del Panel

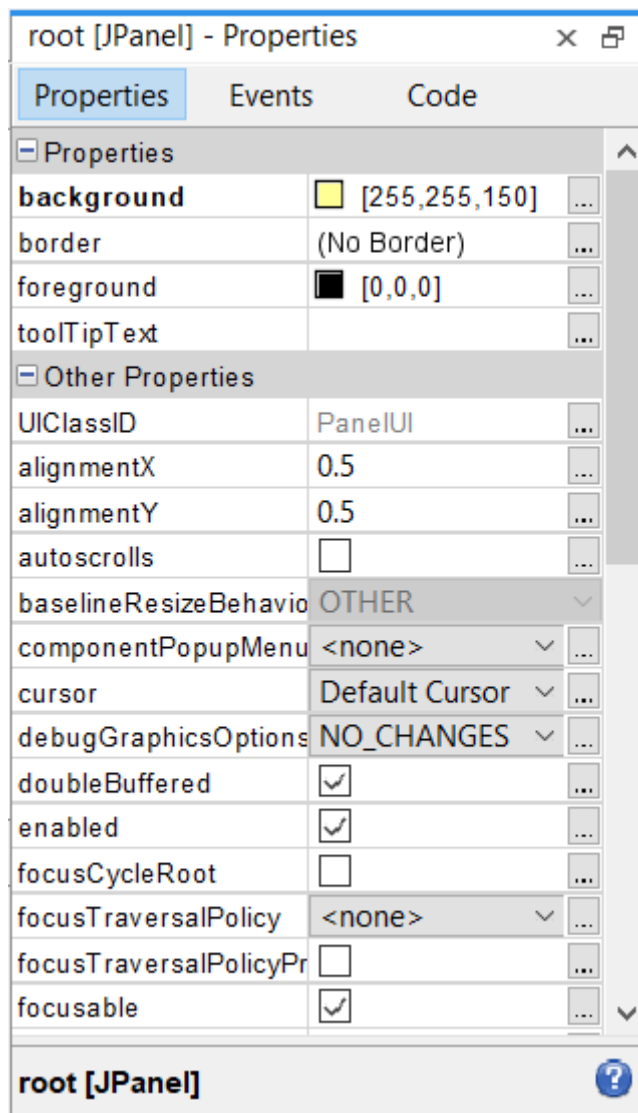


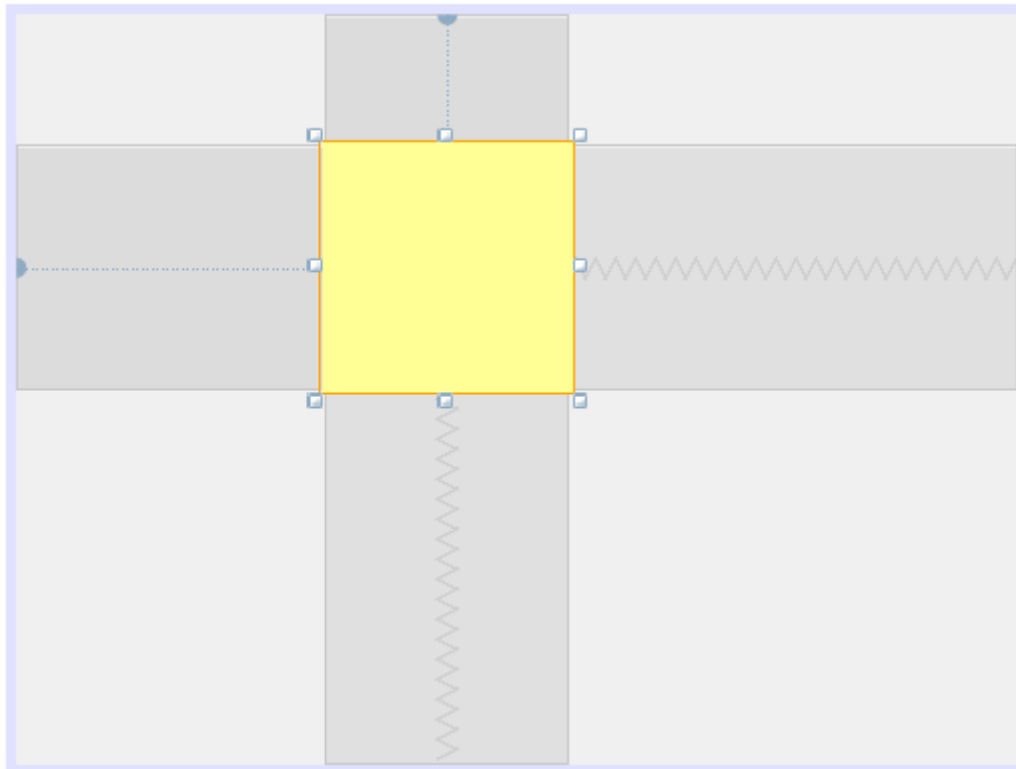


Poner fondo de color al Panel “root”

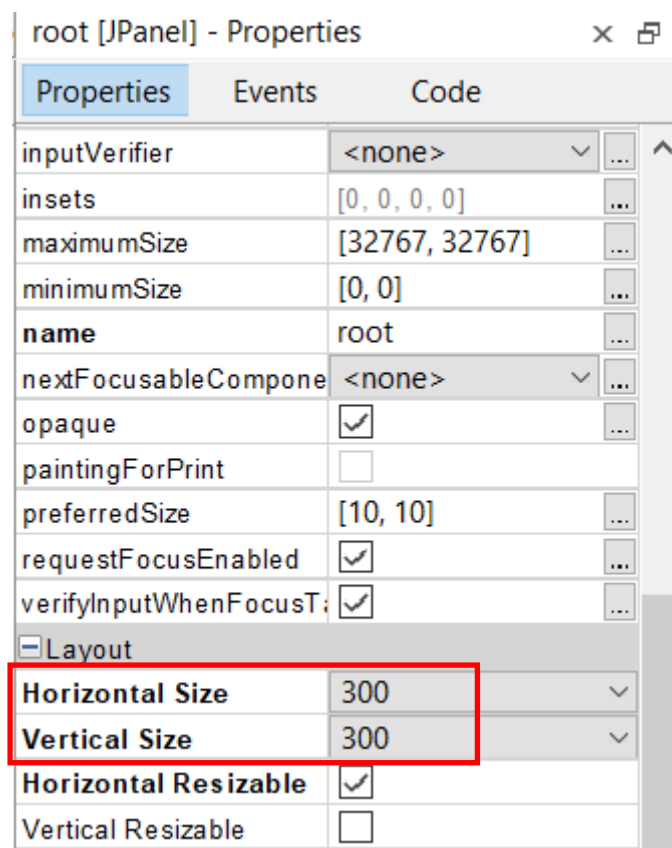
- Pinchar sobre la propiedad background

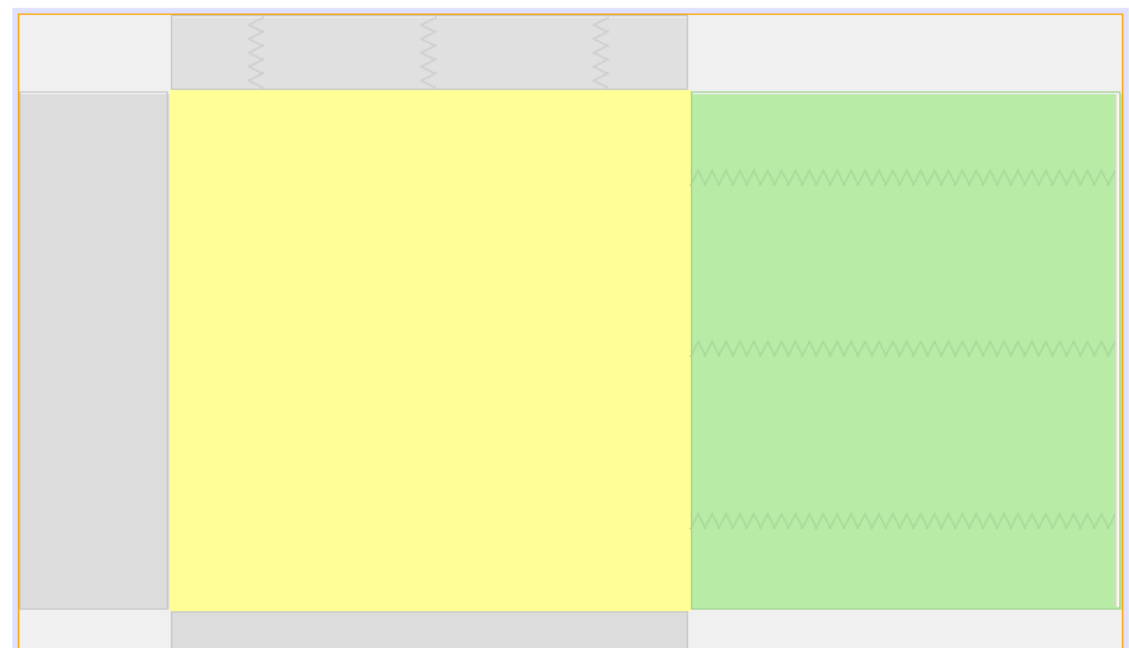






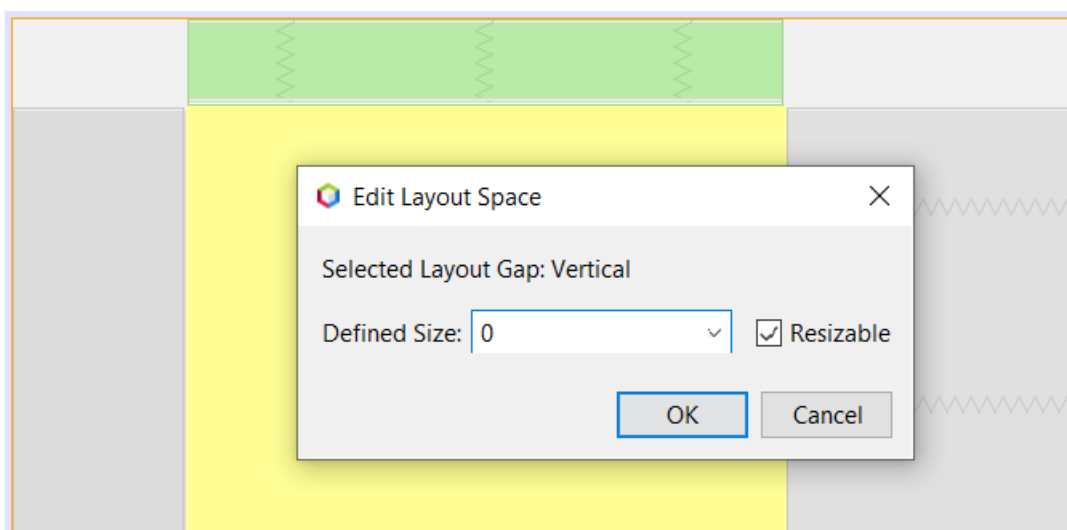
Establecer el ancho y alto del panel

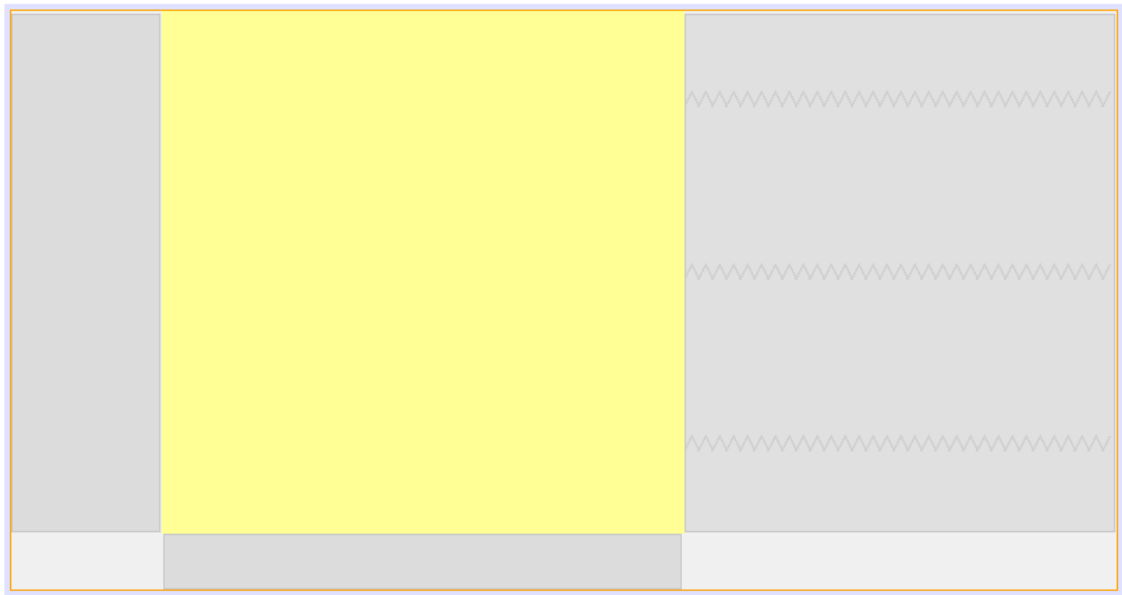




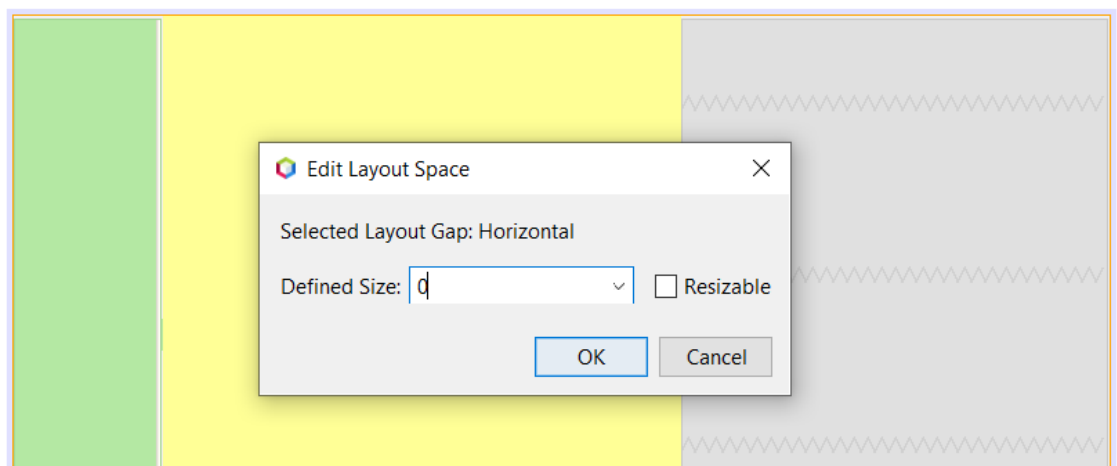
Quitar el margen del panel

- Quitar el margen superior (pulsar doble click sobre el margen superior)

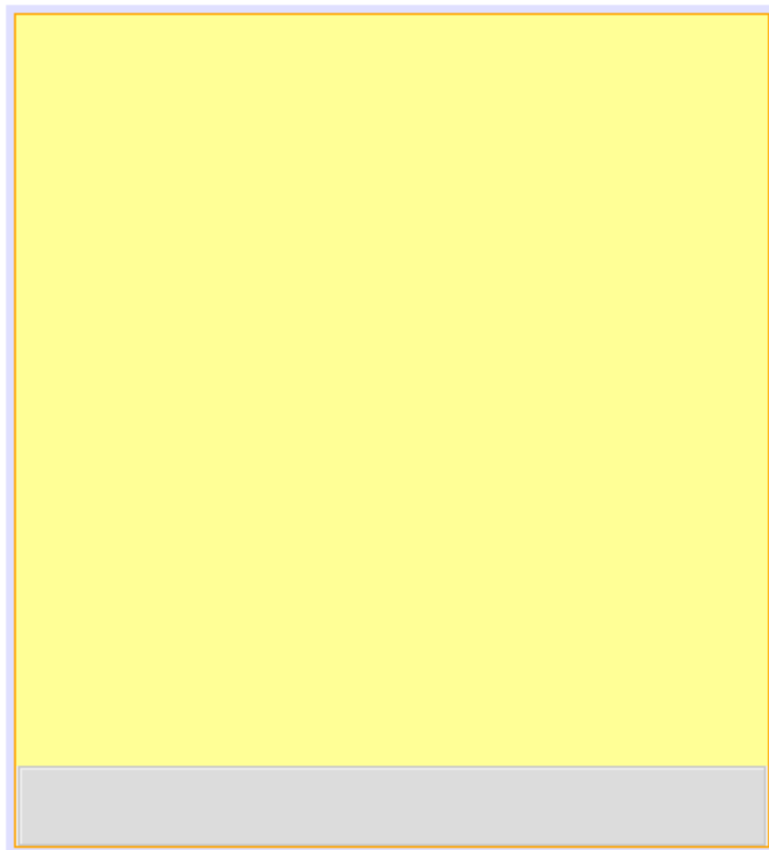
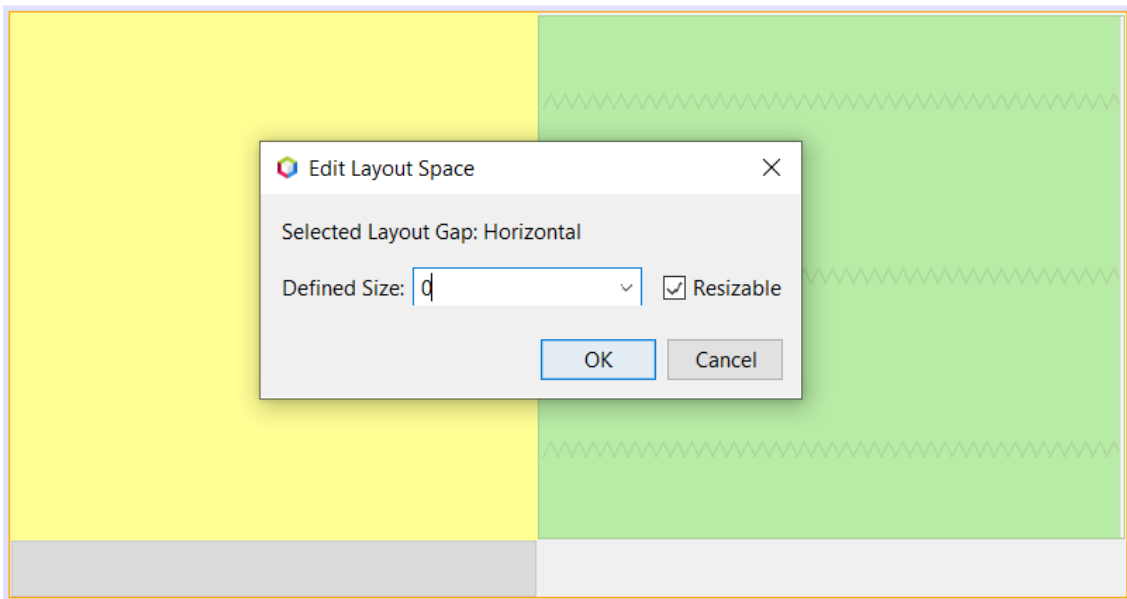




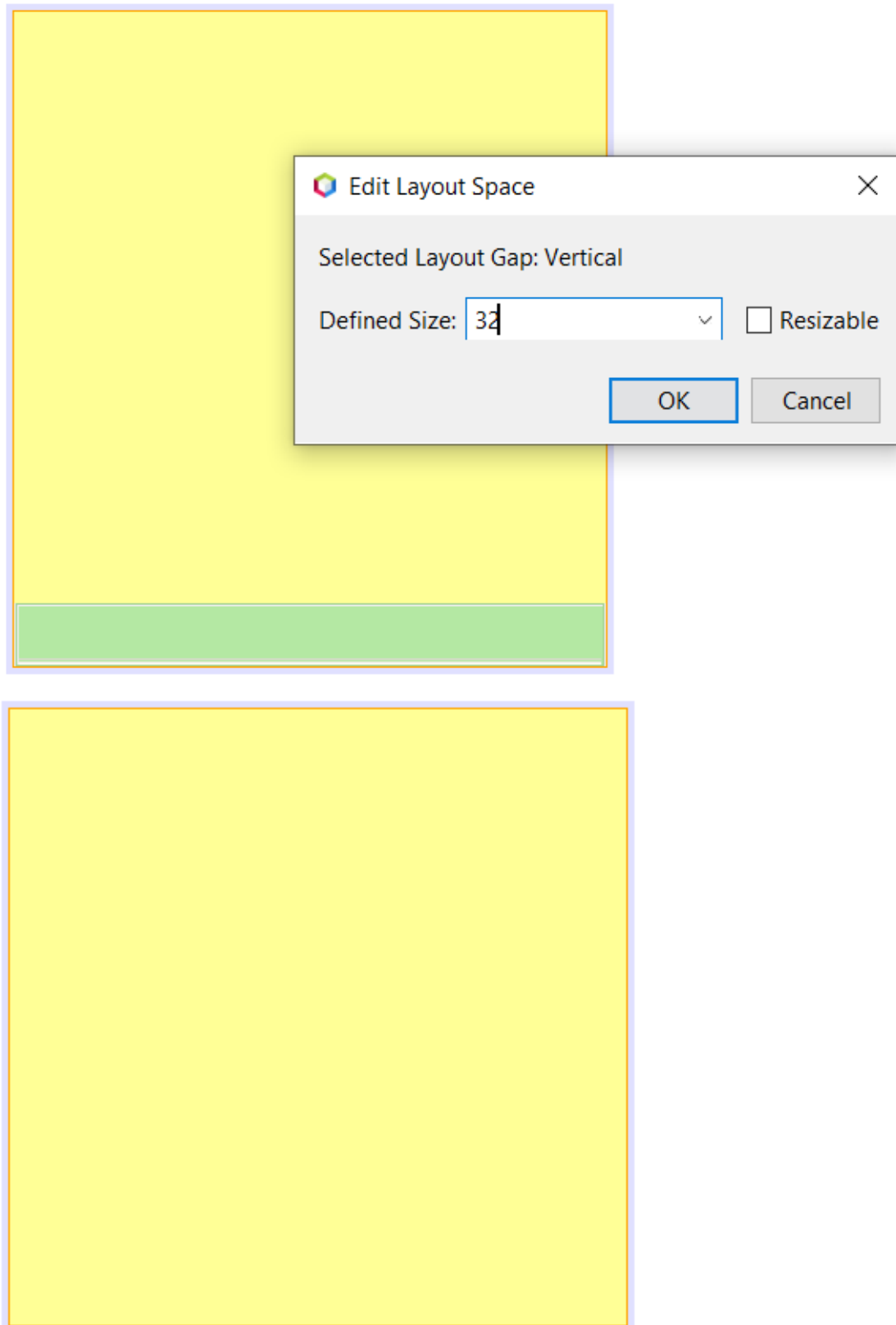
- Quitar el margen izquierdo (pulsar doble click sobre el margen izquierdo)



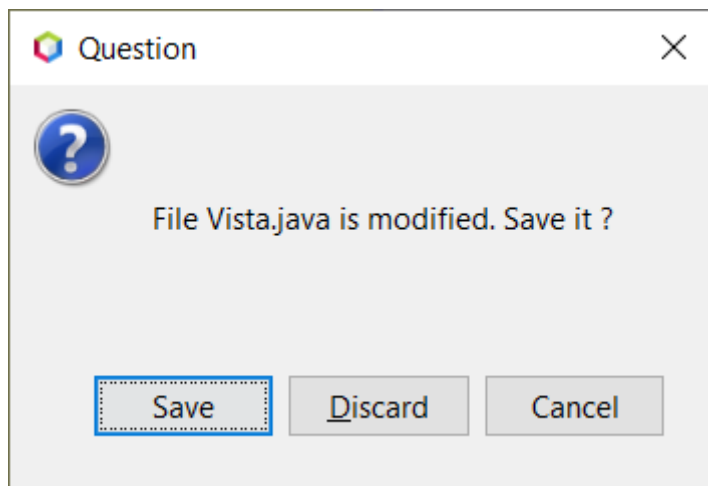
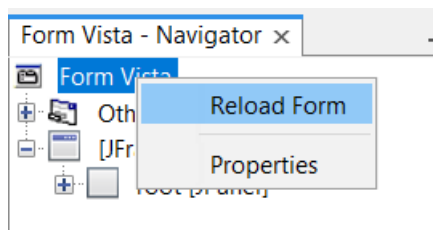
- Quitar el margen derecho (pulsar doble click sobre el margen derecho)



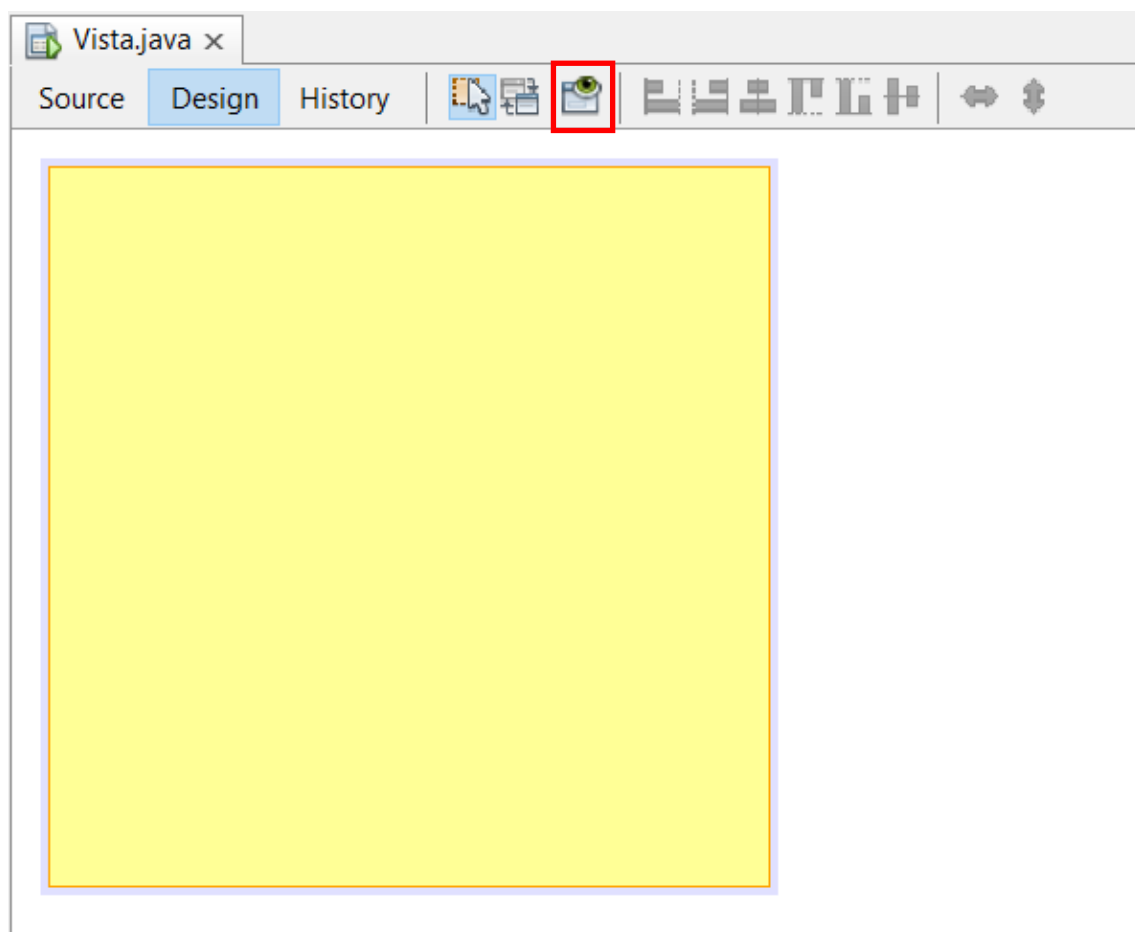
- Quitar el margen inferior (pulsar doble click sobre el margen inferior)

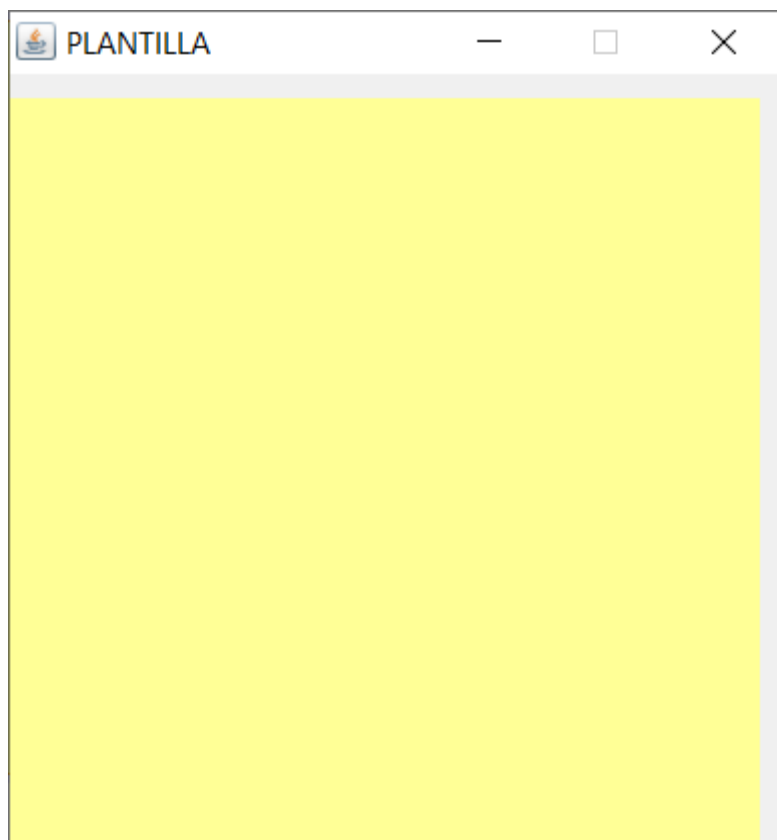


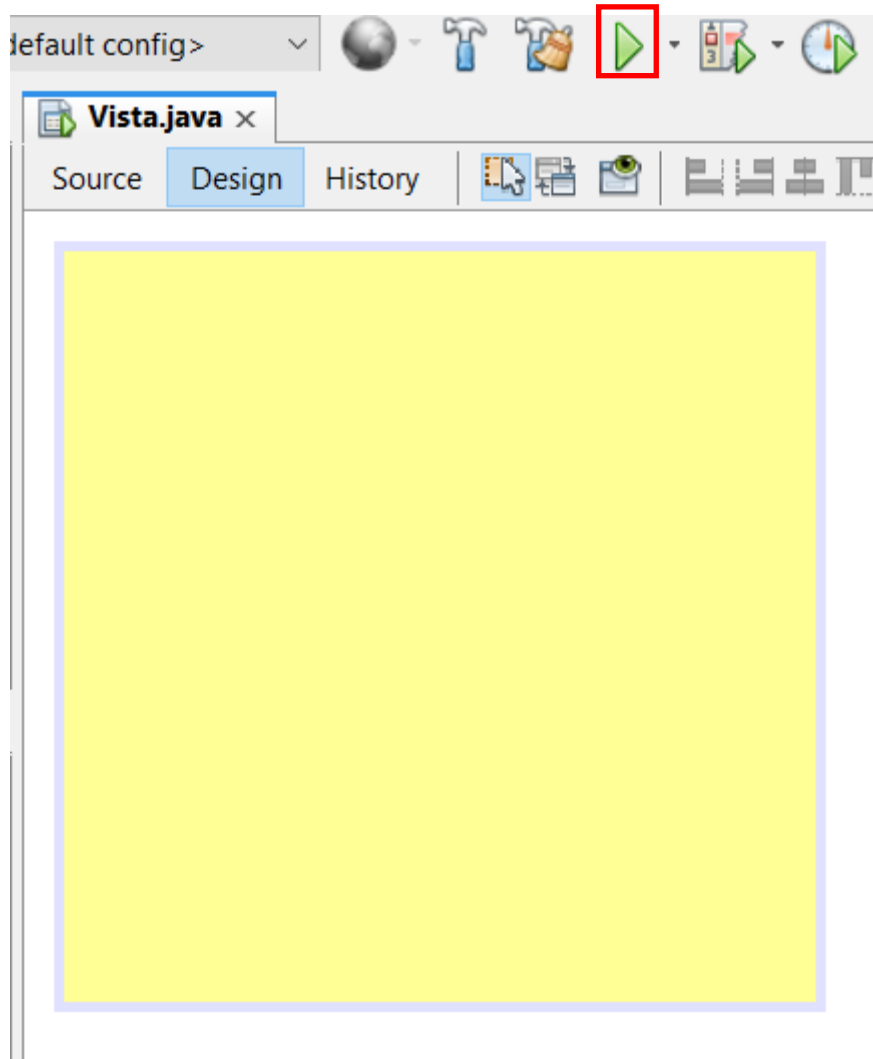
- Recargar el formulario

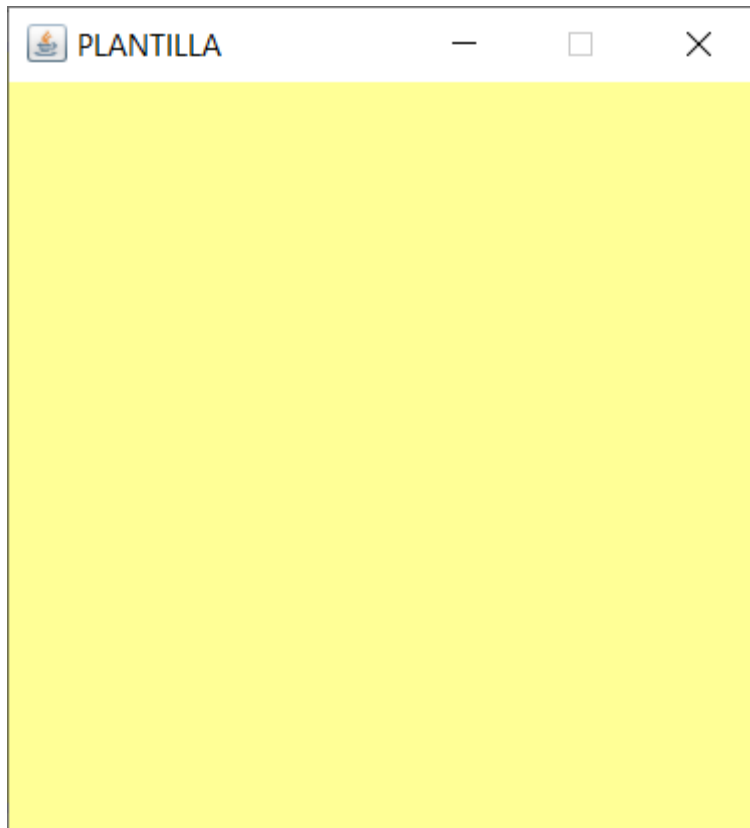


- Mostrar el formulario en diseño









Codigo de la vista (pestaña Source)

A screenshot of the Source tab in an IDE showing Java code for Vista.java. The code is as follows:

```
1  /*  
2   * Click nbfs://nbhost/SystemFileSystem/Template  
3   * Click nbfs://nbhost/SystemFileSystem/Template  
4   */  
5  package vistas;  
6  
7  /**  
8   *  
9   * @author luiss  
10  */  
11  public class Vista extends javax.swing.JFrame {  
12
```

Se pueden quitar los comentarios que están en rojo

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt  
to change this license  
 * Click nbfs://nbhost/SystemFileSystem/Templates/GuiForms/JFrame.java to edit  
this template  
*/
```

```

package vistas;

/**
 *
 * @author luiss
 */
public class Vista extends javax.swing.JFrame {

    /**
     * Creates new form Vista
     */
    public Vista() {
        initComponents();
    }

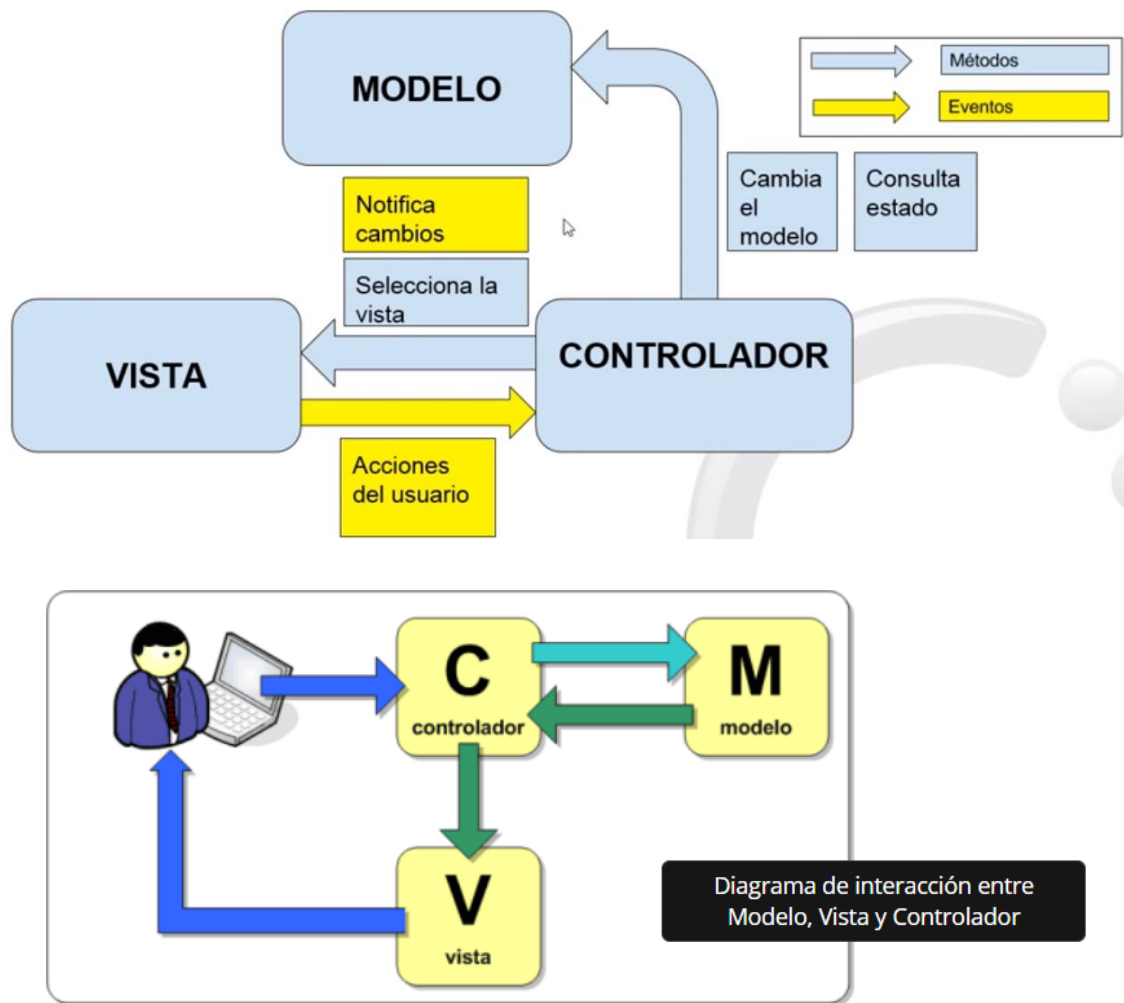
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Vista().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JPanel root;
    // End of variables declaration
}

```

MODELO VISTA CONTROLADOR (MVC)



En el Controlador se encuentran los componentes capaces de procesar las interacciones del usuario, consultar o actualizar el Modelo, y seleccionar las Vistas apropiadas en cada momento.

En la Vista encontraremos los componentes responsables de generar la interfaz con el exterior, por regla general, aunque no exclusivamente, el UI de nuestra aplicación.

El Modelo contiene principalmente las entidades que representan el dominio, la lógica de negocio, y los mecanismos de persistencia de nuestro sistema.

<https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx>

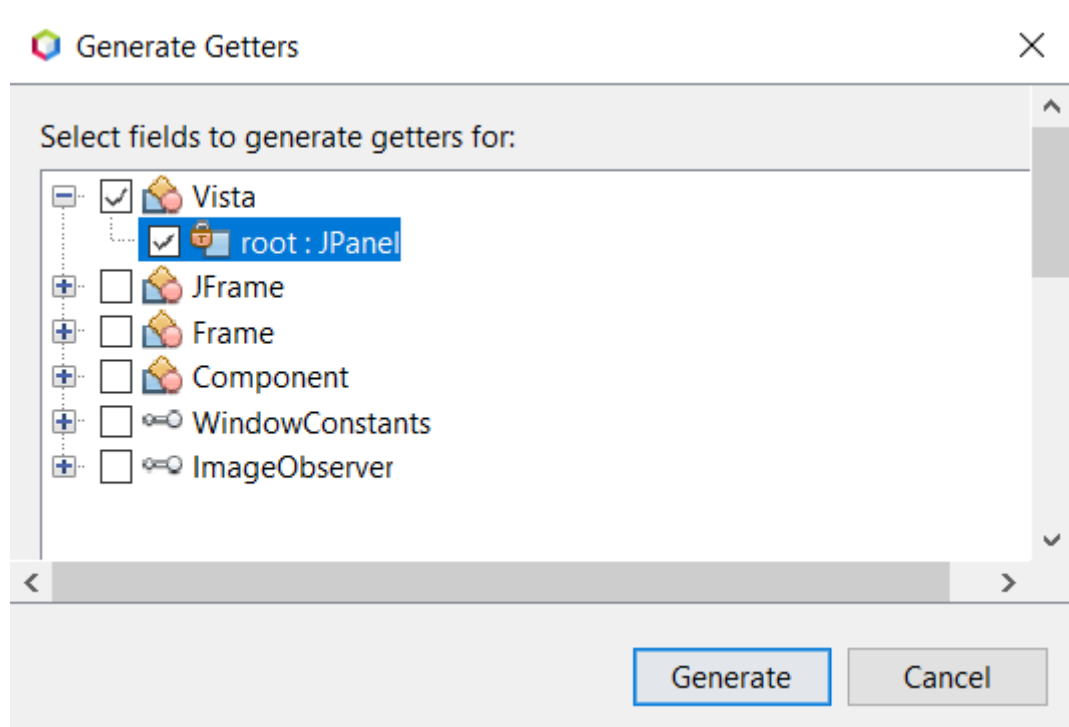
VISTA

Se trata de la clase anterior que está definida en el paquete “vistas” .

En dicha clase se define:

- La interfaz gráfica del usuario
- Se definen los métodos get asociados a cada control
- Se agrupan controles en array y se definen los métodos get de cada componente y un método que obtiene el número de componentes.

Por ejemplo, dentro de la clase Vistas se define el método getRoot asociado al Panel “root”:



```
public JPanel getRoot() {  
    return root;  
}
```

MODELO

Clase definida en el paquete “modelos”

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Superclass:

Interfaces:

< Back Next > **Finish** Cancel Help

```
package modelos;
public class Modelo {

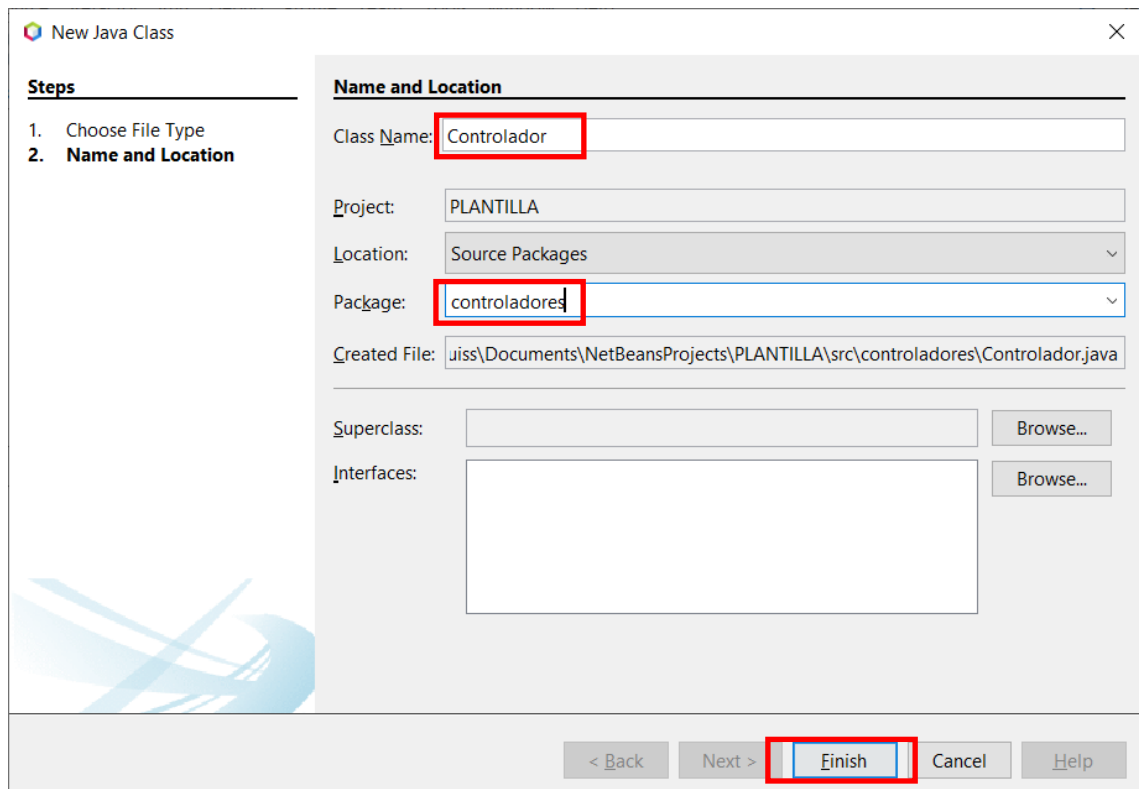
}
```

Contendrá la siguiente información:

- Datos de salida que se mostrarán en la interfaz gráfica después de realizar un cálculo más o menos complejo, un acceso a base de datos o un acceso a un fichero.
- Rutas a recursos.

CONTROLADOR

Clase definida en el paquete “controladores”:



```
package controladores;  
  
public class Controlador {  
  
}
```

El controlador se encarga de:

- Acceder al modelo y a la vista
- Modificar la vista inicial (datos de validación, etc)
- Añadir a un control de la vista un escuchador con su correspondiente controlador
- Mover datos a la vista desde el modelo
- Extraer datos de la vista para pasárselos al modelo.

El código del Controlador se modifica del siguiente modo:

```
package controladores;  
  
import modelo.Modelo;  
import vistas.Vista;  
  
public class Controlador {  
    private Modelo m;  
    private Vista v;
```

```

public Controlador(Modelo m, Vista v)
{
    this.m = m;
    this.v = v;

    //Modificaciones de la vista

    //Añadir escuchadores

    //Acciones
}

/* METODOS PRIVADOS */

/* ***** */
/* EVENTOS */
/* ***** */
}

```

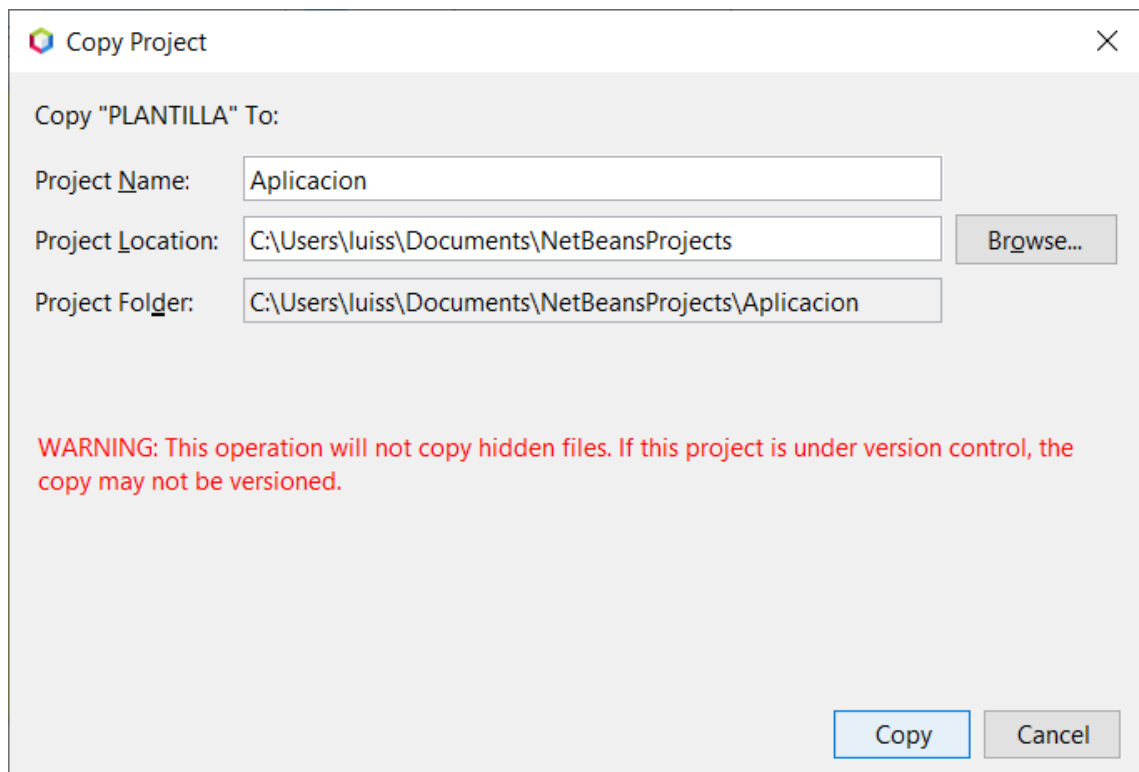
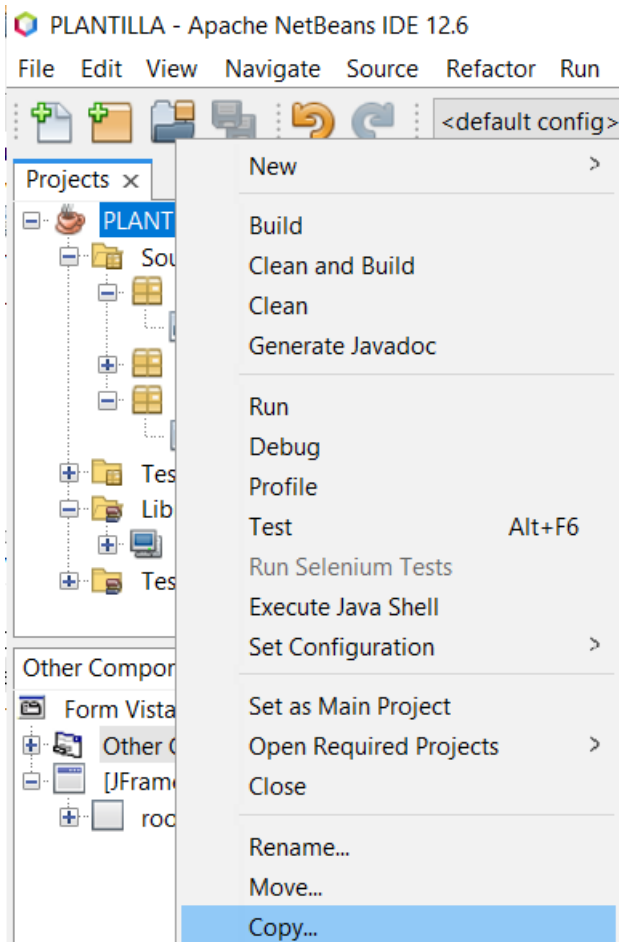
El método “run” de la clase Vista se modifica de la siguiente manera:

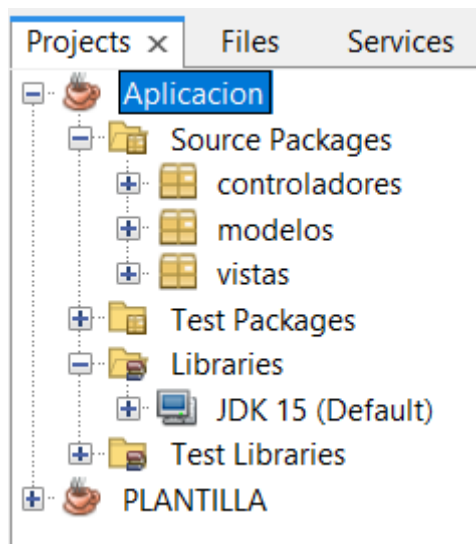
```

public void run() {
    Vista v = new Vista();
    modelos.Modelo m = new modelos.Modelo();
    controladores.Controlador c = new controladores.Controlador(m,v);
    v.setVisible(true);
}

```

CREAR UNA APLICACIÓN (a partir de la PLANTILLA)





FORMULARIOS BÁSICOS

- Contenedor de tipo panel → JPanel
- Contenedor de tipo panel con fondo de imagen → JPanelBackground
- Controles de cajas de texto de solo lectura → JTextField
- Controles de caja de texto de escritura → JText
- Controles de etiquetas → JLabel
- Controles de botones → JButton
- Arrays de controles

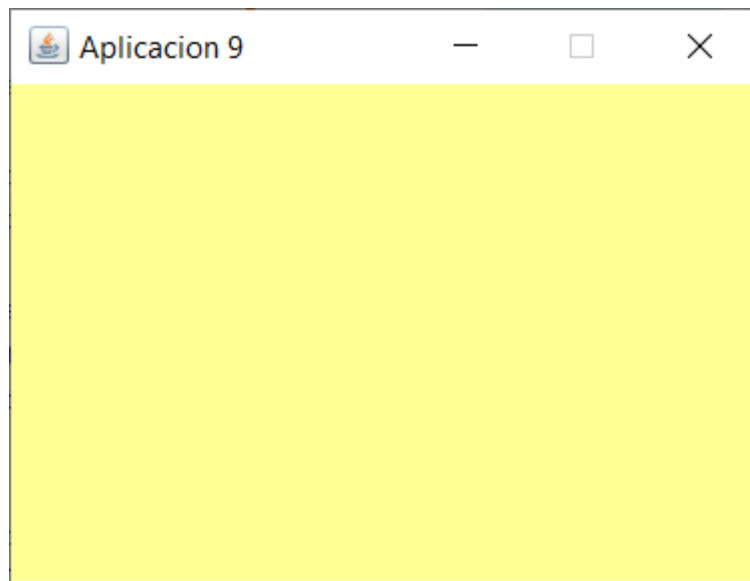
Propiedades genéricas del JFrame

EVENTOS	
WindowListener	Lanza los eventos de la ventana <ul style="list-style-type: none"> • windowLostFocus • windowLostFocus • windowStateChanged • windowDeactivated • windowActivated • windowDeiconified • windowIconified • windowClosed • windowClosing • windowOpened

PROPIEDADES	
title	Título del formulario
defaultCloseOperation	Operación por defecto que se realiza al cerrar el formulario (nada, salir, cerrarse, ocultar)
iconImage	Icono del formulario
layout	Establece el tipo de gestor para disponer los controles en el formulario.
locationRelativeTo	Posicionar el formulario en la pantalla
resizable	Redimensionar o no el formulario
undecorated	Quitar la barra principal del formulario con el icono, título y botones de maximizar, minimizar y cerrar

JPanel “root”


1. Hágase un formulario con el título de “Aplicación 9”, no redimensionable, centrado en la pantalla. Dicho formulario contendrá un panel de 300x200 con un fondo de color 255,255,150.



Propiedades del marco (en todos los formularios)

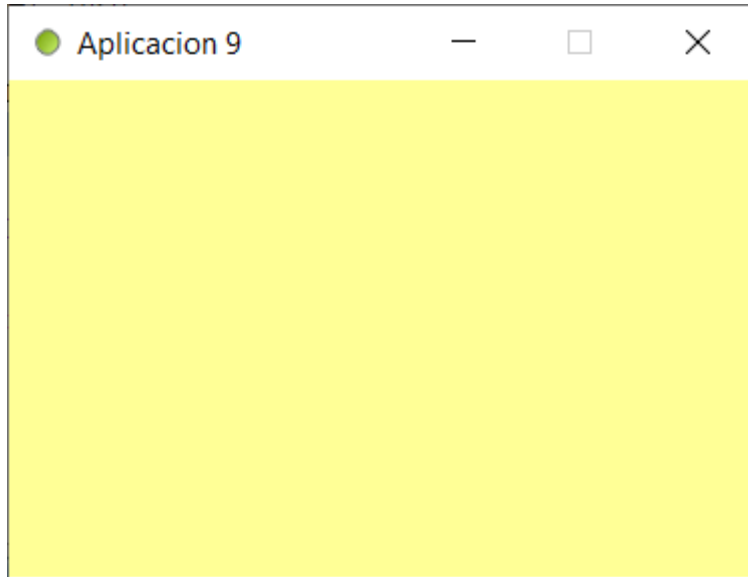
Form Formulario - Properties	
Code Generation	
Variables Modifier	private
Local Variables	<input type="checkbox"/>
Generate Full Classnames	<input type="checkbox"/>
Generate Mnemonics Code	<input type="checkbox"/>
Listener Generation Style	Anonymous Inner Classes
Layout Generation Style	Standard Java 6 code
Resources and Internationalization	
Set Component Names	<input checked="" type="checkbox"/>
Automatic Internationalization	<input type="checkbox"/>
Properties Bundle File	formularios/Bundle
Design Locale	

Vista

JFrame	
defaultCloseOperation	EXIT_ON_CLOSE
title	Aplicacion 9
resizable	<input type="checkbox"/>
Generate Center	<input checked="" type="checkbox"/>
JPanel	
background	 [255,255,150]
Horizontal Size	300
Vertical Size	200
Variable Name	root

Icono del formulario

1b Hágase un formulario con el título de “Aplicación 9”, no redimensionable, centrado en la pantalla. Dicho formulario contendrá un panel de 300x200 con un fondo de color 255,255,150 y como icono el de “verde.png”



Modelo

```
final static public URL ICONO =
Modelo.class.getResource("/rec/verde.png");

static public Image cargar(URL url)
{
    return new ImageIcon(url).getImage();
}
```

Vista

JFrame	
title	Aplicacion 9
defaultCloseOperation	DO_NOTHING
JPanel	
Horizontal Size	300
Vertical Size	200

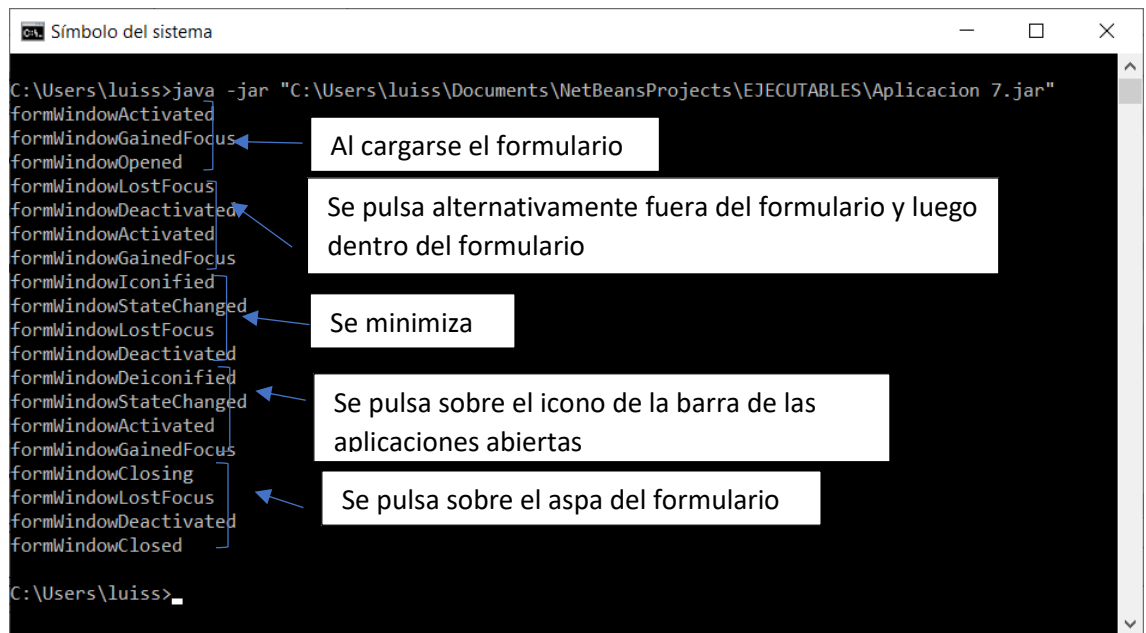
Controlador

(en el constructor)

```
//Modificar la vista
v.setIconImage (Modelo.cargar (Modelo.ICONO) );
```

Eventos de la ventana

1c Modifíquese el formulario anterior con la operación por defecto de cierre “DISPOSE” y que muestre por consola para cada uno de los eventos del formulario un mensaje por la consola con el evento que se ha lanzado.



Controlador

```
(en el constructor)
//Añadir escuchadores
v.addWindowListener(new eventosVentana());

(en la clase)
/* EVENTOS DE LA VENTANA */
//private class eventosVentana implements WindowListener {
private class eventosVentana extends WindowAdapter {

    @Override
    public void windowLostFocus(WindowEvent e) {

    }

    @Override
    public void windowGainedFocus(WindowEvent e) {

    }

}
```



```

@Override
public void windowStateChanged(WindowEvent e) {

}

@Override
public void windowDeactivated(WindowEvent e) {

}

@Override
public void windowActivated(WindowEvent e) {

}

@Override
public void windowDeiconified(WindowEvent e) {

}

@Override
public void windowIconified(WindowEvent e) {

}

@Override
public void windowClosed(WindowEvent e) {

}

@Override
public void windowClosing(WindowEvent e) {

}

@Override
public void windowOpened(WindowEvent e) {

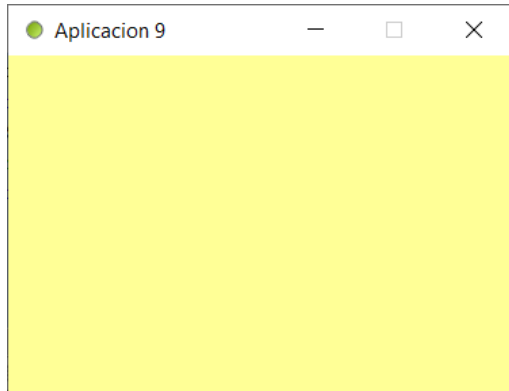
}
}

```

Cancelación del cierre del formulario y mover de datos del modelo a la vista

1d Modifíquese el formulario anterior de modo que la operación por defecto de cierre sea “DO_NOTHING” y que sólo se cierre cuando al generar un número aleatorio entre 0 y 9, ambos inclusive, distinto del último generado, se obtenga el

valor de 0. Cada vez que se genera un número aleatorio (incluido el 0) se muestra como título del formulario. Para que de tiempo a ver el valor de 0, antes de cerrar el formulario, se dormirá la aplicación durante 300 milisegundos (utilizar el método estático “sleep” de la clase “java.lang.Thread”)



Carga inicial



Al pulsar una vez sobre X

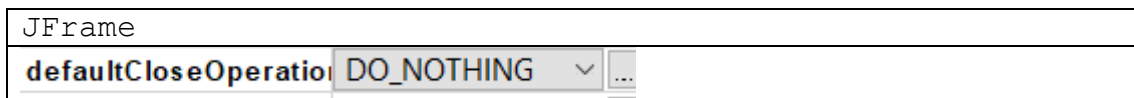


Al pulsar de nuevo sobre X



Al pulsar de nuevo sobre X (300 ms y se cierra)

Vista



Modelo

```
private int numero = -1;
public int getNumero()
{
    return numero;
}

public void aleatorio()
{
    int n = 0;
```

```

        while(true)
        {
            n = (int) (Math.random()*10);
            if(n!=numero) break;
        }

        numero = n;
    }

    public boolean numeroEsCero() {
        return numero==0;
    }

```

Controlador

```

(en el constructor)
//Datos a mover a la vista
m.addPropertyChangeListener(Modelo.PROP_NUMERO, e->{
    int numero = m.getNumero();
    v.setTitle(String.valueOf(numero));
});

/* EVENTOS DE LA VENTANA */
private class eventosVentana extends WindowAdapter {
    ...
    @Override
    public void windowClosing(WindowEvent e) {
        ...
        m.aleatorio();
        int numero = m.getNumero();
        v.setTitle(String.valueOf(numero));

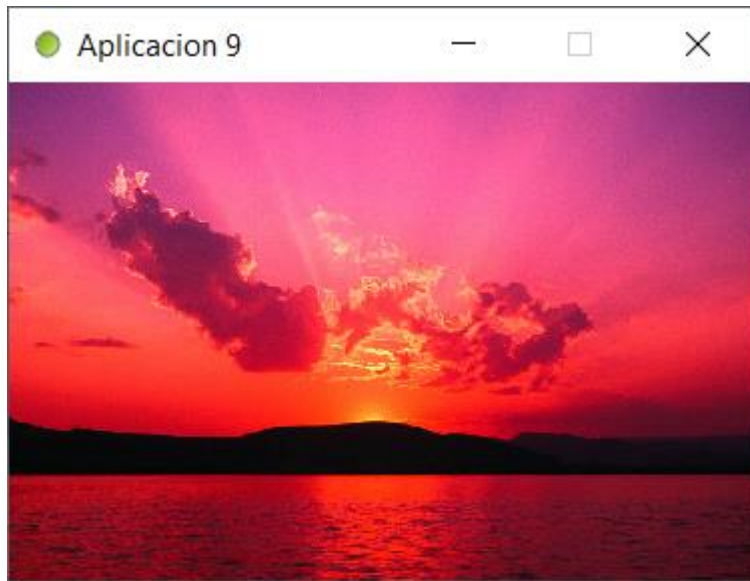
        if(m.numeroEsCero())
            try {
                Thread.sleep(300);
                v.dispose();
            } catch (InterruptedException ex) {}
    }
}

```

Creación de un control JPanel con fondo de imagen (JPanelBackground)

PROPIEDADES	
image	Establece la imagen de fondo

1e Modifíquese el formulario anterior de modo que el panel “root” tenga como fondo de imagen la de “puesta.jpg”.



Clase “clases.JPanelBackground”

```
public class JPanelBackground extends JPanel implements
Serializable {
    private Image image;

    public Image getImage() {
        return image;
    }

    public void setImage(Image image) {
        this.image = image;
    }

    @Override
    protected void paintComponent(Graphics g) {
        g.drawImage(image, 0, 0, getWidth(), getHeight(),
this);
        this.repaint();
    }
}
```

Modelo

```
final static public URL PUESTA =
Modelo.class.getResource("/rec/puesta.jpg");
```

Vista

- Quitar el panel actual
- Arrastrar el archivo JPanelBackground.java al formulario.

JPanelBackground	
Horizontal Size	300
Vertical Size	200
Variable Name	root

Controlador

(en el constructor)

//Modificar la vista

```
v.getRoot().setImage(Modelo.cargar(Modelo.PUESTA));
```

Propiedades y eventos genéricos de la mayoría de los controles y contenedores

EVENTOS	
FocusListener	Se lanza cuando un control recibe o pierde el foco <ul style="list-style-type: none"> gainedFocus lostFocus

PROPIEDADES	
background	Color del fondo
cursor	Forma del cursor al pasar sobre el control
enabled	Habilitar/deshabilitar
focusable	Recibir o no el foco con el tabulador
font	Fuente
foreground	Color del texto
toolTipText	Cartelito

JButton. Cálculo de datos de salida en el controlador.

EVENTOS	
ActionListener	Se lanza al pulsar el botón

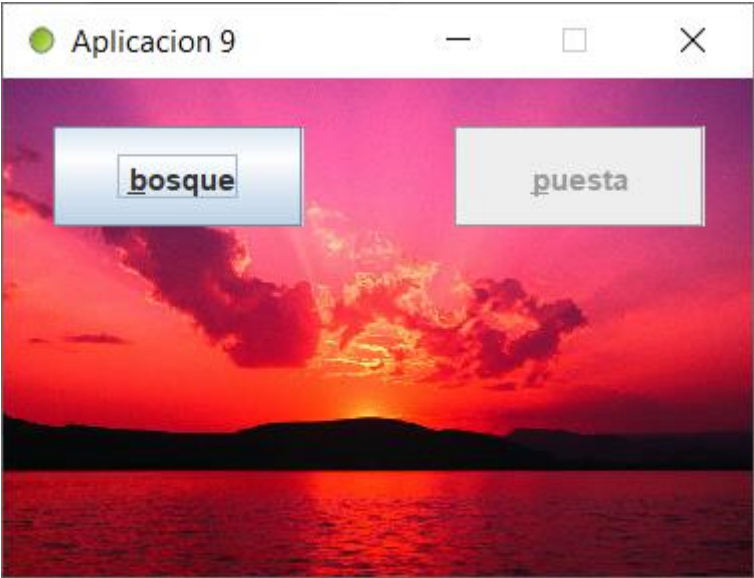
PROPIEDADES	
text	Texto actual del botón
icon	Icono del botón
mnemonic	Tecla de acceso directo

METODOS	
doClick()	Lanza el evento “click” desde código

METODOS del panel principal del formulario	
setDefaultButton()	Asociar la tecla “Enter” al evento click de un botón, por ejemplo “incrementar”

	<code>this.getRootPane().setDefaultButton(incrementar);</code>
--	--

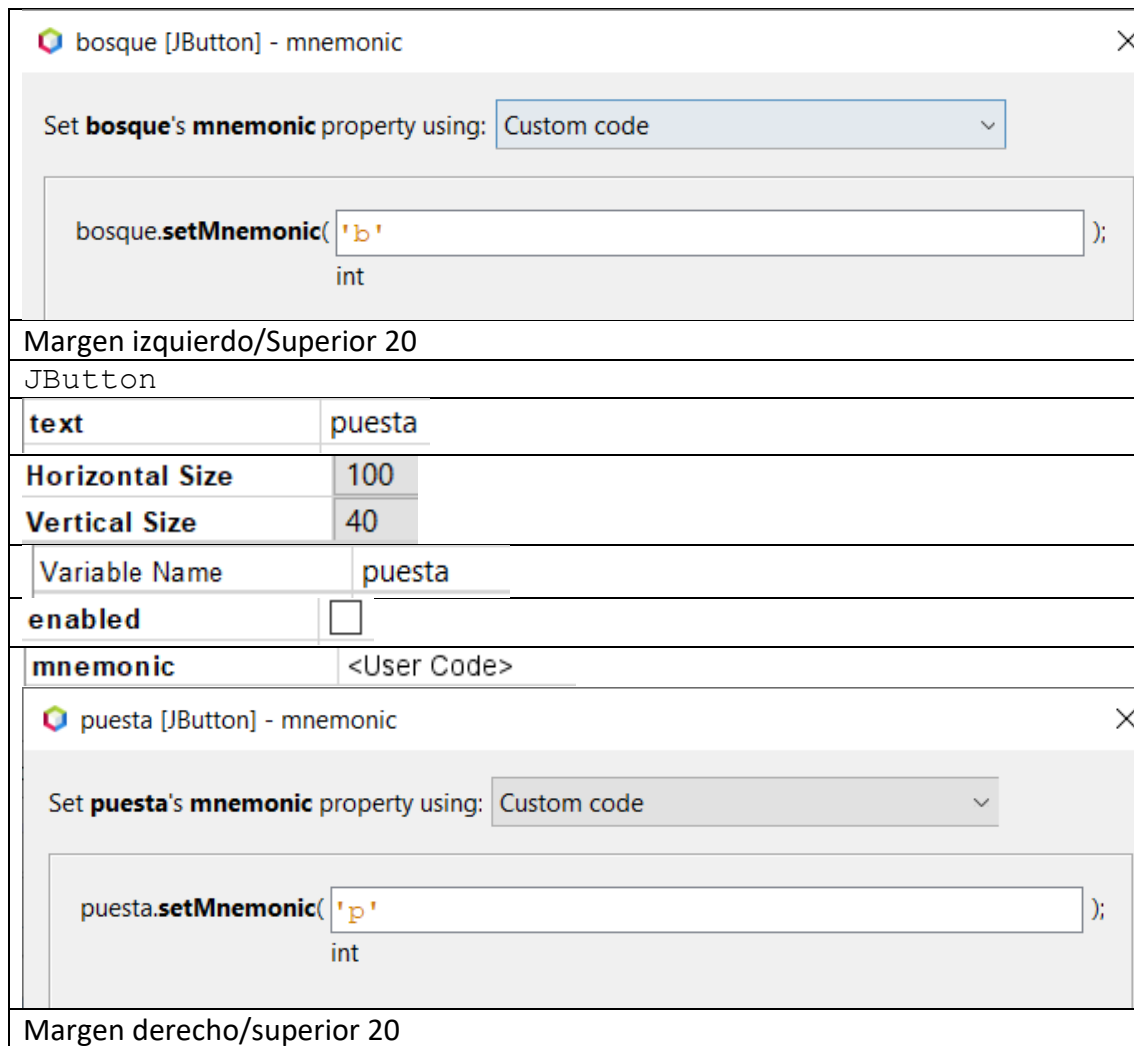
1f Modifíquese el formulario anterior de modo que vaya intercambiando la imagen de fondo del al pulsar alternativamente sendos botones que se deshabilitan/habilitan mutuamente.



- a) Los botones tendrán un tamaño de 100 x 40
- b) El botón de “puesta” está separado de la parte superior 20 y de la parte derecha 20
- c) El botón de “bosque” está separado de la parte superior 20 y de la parte izquierda 20

Vista

JButton	
text	bosque
Horizontal Size	100
Vertical Size	40
Variable Name	bosque
mnemonic	<User Code>



Modelo

```
final static public URL BOSQUE =
Modelo.class.getResource("/rec/bosque.jpg");
```

Controlador

```
(en el constructor)
//Añadir escuchadores
v.getBosque().addActionListener(this::mostrarBosque);
v.getPuesta().addActionListener(this::mostrarPuesta);

(en la clase)
/* EVENTO bosqueClick */
private void mostrarBosque(ActionEvent e) {
    v.getRoot().setImage(Modelo.cargar(Modelo.BOSQUE));
    v.getBosque().setEnabled(false);
    v.getPuesta().setEnabled(true);
}
```

```

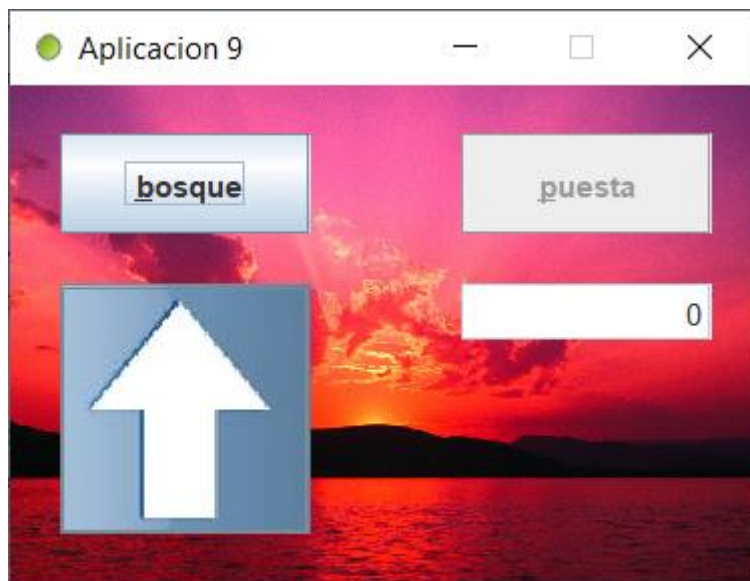
/* EVENTO puestaClick */
private void mostrarPuesta(ActionEvent e) {
    v.getRoot().setImage(Modelo.cargar(Modelo.PUESTA));
    v.getPuesta().setEnabled(false);
    v.getBosque().setEnabled(true);
}

```

JButton gráfico y JTextField de sólo lectura. Cálculos sencillos realizados en el controlador.

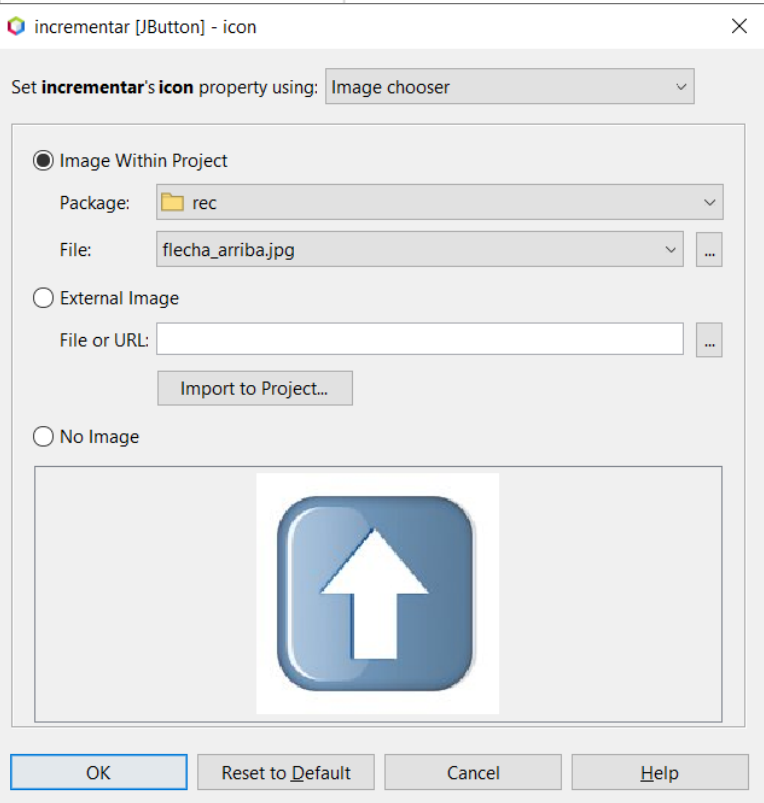
PROPIEDADES	
text	Texto actual de la caja de texto
editable	De lectura o escritura
columns	Número de columnas preferible a mostrar
horizontalAlignment	Alineación horizontal

1g Modifíquese el formulario anterior de forma que se añada un botón gráfico y una caja de texto. Al pulsar el botón se incrementará en uno el contenido de la caja de texto.



- Un botón “incrementar” de 100 x 100 con el icono “flecha_arriba.jpg” con una separación superior e izquierda de 20 y que se ejecutará al pulsar la tecla ENTER.
- Una caja de texto “valor” de 100 x 22 de sólo lectura y con una separación superior y derecha de 20 que no puede recibir el foco con el tabulador y con 8 columnas.
- Al pulsar el botón se incrementará en una unidad el contenido de la caja de texto.

Vista

JButton	
text	
Horizontal Size	100
Vertical Size	100
Variable Name	incrementar
icon	flecha_arriba.jpg
	
Margen izquierdo/Superior 20	
JTextField	
editable	<input type="checkbox"/>
background	<input type="checkbox"/> [255,255,255]
horizontalAlignment	RIGHT
text	0
Horizontal Size	100
Vertical Size	22
focusable	<input type="checkbox"/>
columns	8
Variable Name	valor

(en el constructor)

```
this.getRootPane().setDefaultButton(incrementar);
```

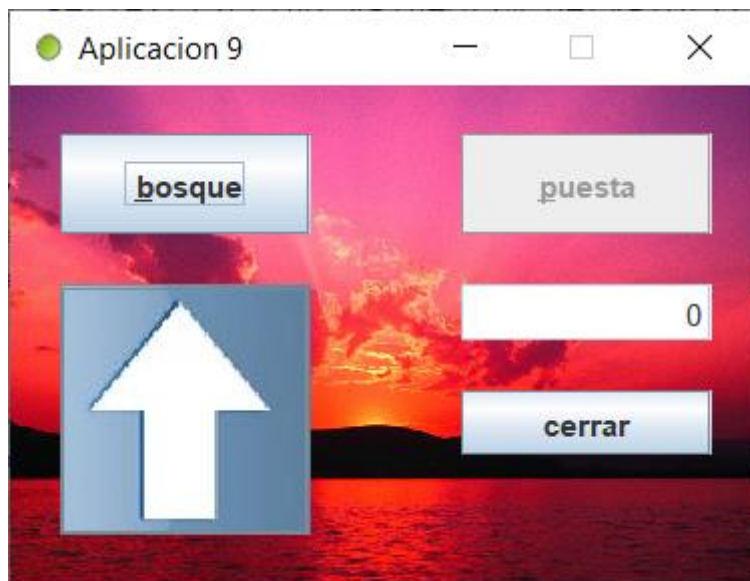
Controlador

```
(en el constructor)
//Añadir escuchadores
v.getIncrementar().addActionListener(this::incrementar
Valor);
```

```
(en la clase)
/* EVENTO v.incrementar Click */
private void incrementarValor(ActionEvent e) {
    int n = Integer.valueOf(v.getValor().getText());
    n++;
    v.getValor().setText(String.valueOf(n));
}
```

Lanzar eventos desde código

1h Modifíquese el formulario anterior de modo que se añada un botón que al pulsarlo se lance desde código el click del botón “incrementar” y el evento “closing” de la ventana.



- d) El botón tendrá un tamaño de 100 x 25 y estará separado superiormente y por la derecha 20.

Vista

JButton	
text	cerrar
Horizontal Size	100
Vertical Size	25
Variable Name	cerrar
Margen derecho/superior 20	

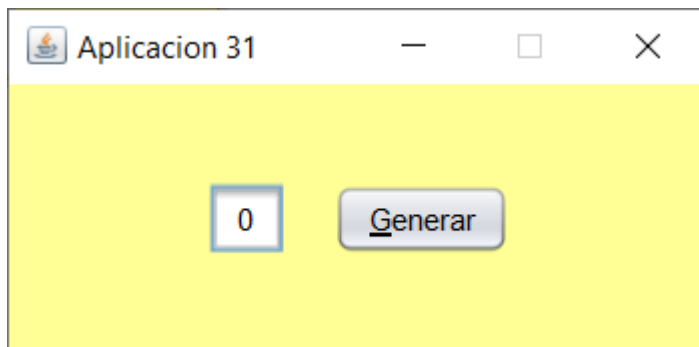
Controlador

```
(en el constructor)
//Añadir escuchadores
v.getCerrar().addActionListener(this::incrementarCerrar);

(en la clase)
/* EVENTO cerrar CLICK */
private void incrementarCerrar(ActionEvent e) {
    v.getIncrementar().doClick();
    v.getWindowListeners()[0].windowClosing(new
WindowEvent(v,WindowEvent.WINDOW_CLOSING));
}
```

EJERCICIO

2. Hágase un formulario que al pulsar un botón genere un número aleatorio entre 0 y 9, ambos inclusive, que se muestra en una caja de texto de solo lectura. Cada vez que se pulsa el botón se genera un número distinto del que hubiera actualmente.



- a) El formulario estará centrado, no será redimensionable y contendrá un panel principal que tendrá un tamaño de 290 x 105 y con un fondo de color 255, 255 y 150.
- b) La caja de texto tendrá un tamaño de 30 x 25, de sólo lectura con fondo blanco y el texto estará centrado. Está separado de la parte superior 40 y de la parte izquierda 80
- c) Un botón de tamaño 80 x 25 separado de la parte superior 40 y de la parte derecha 40.

Formularios sin iconos, título, botones de maximizar, minimizar y cierre

EJERCICIO (salvo quitar la barra del formulario y dar un borde al panel)

3. Hágase un formulario sin decorar. Al pulsar un botón el formulario se cerrará. (Aplicacion12)

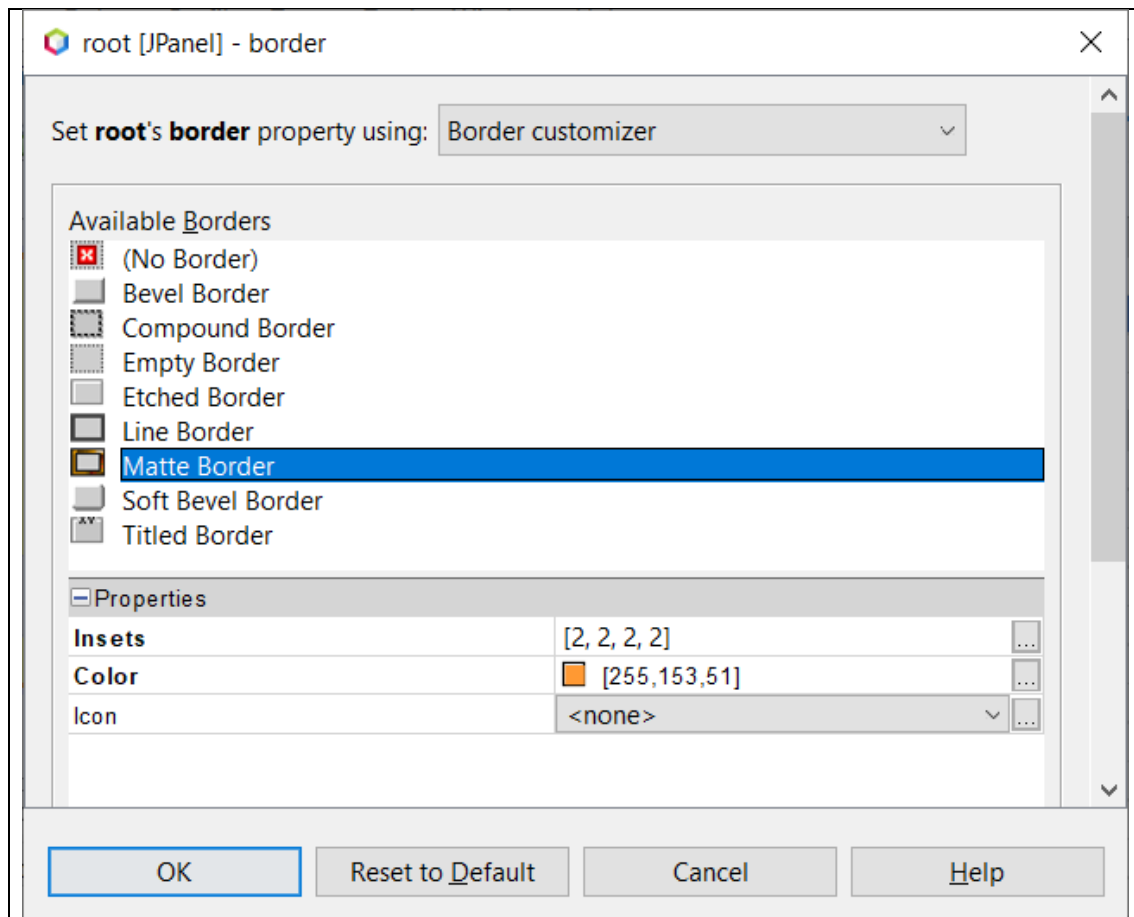


El panel principal tendrá un tamaño de 300 x 200 con un borde MatteBorder de 2 pixeles y con el color 255, 153, 51. El panel estará centrado y no será redimensionable y contendrá los siguientes controles:

- a) Un botón “cerrar” de 80 x 40 con un fondo de color blanco y con una separación horizontal derecha y vertical inferior de 20

Vista

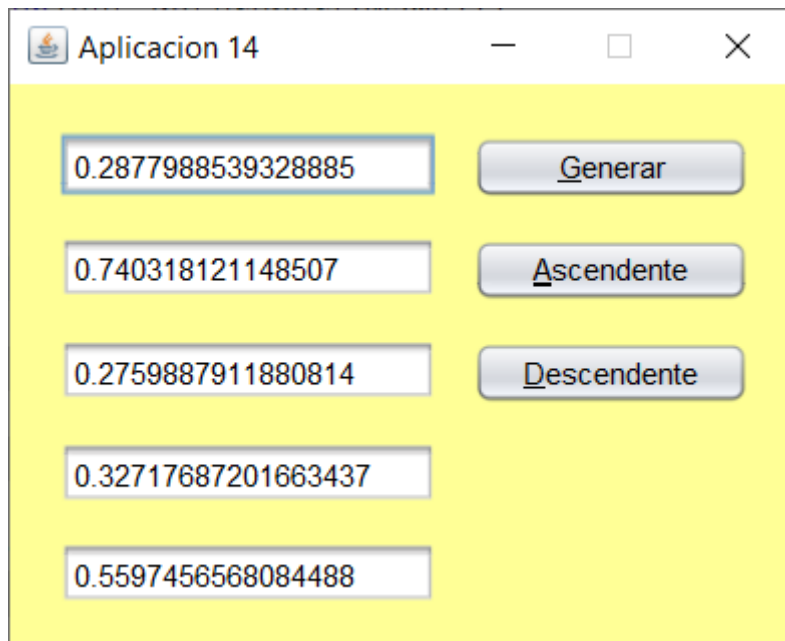
JFrame	
undecorated	<input checked="" type="checkbox"/>
JPanel	
border	[MatteBorder]



Arrays de controles

EJERCICIO (sólo el diseño)

4. Hágase un formulario que permita generar 5 números aleatorios reales entre 0.0 y 1.0 (este último sin incluir), ordenarlos ascendentemente y descendentemente al pulsar sendos botones.



- a) El panel principal tiene un tamaño de 315 x 225 con un relleno de 20.
- b) Todas las cajas de texto tienen un tamaño de 150 x 25, son de solo lectura, no pueden recibir el foco con el tabulador, con 15 columnas, se separan entre sí 15 y se separan (las tres primeras) de los botones 15
- c) Todos los botones tienen un tamaño de 110 x 25 y se separan entre sí 15

Vista

(en el constructor)

```
numeros = new
JTextField[]{numero1,numero2,numero3,numero4,numero5};
```

(en la clase)

```
private JTextField[] numeros;

public JTextField getNumeros(int i) {
    return numeros[i];
}

public int lengthNumeros()
{
    return numeros.length;
}
```

Modelo

```
private double[] valores = {0 , 0, 0, 0, 0};
```

```

    public double getValores(int index) {
        return this.valores[index];
    }

    public void generar()
    {
        for(int i=0;i<valores.length;i++)
            valores[i]=Math.random();
    }

    public void ordenarAscendentemente(double[] a)
    {
        Arrays.sort(valores);
    }

    public void ordenarDescendentemente(double[] a)
    {
        double[] a = valores.clone();
        Arrays.sort(valores);
        for(int i=0;i<valores.length/2;i++)
        {
            double v = valores[i];
            valores[i]=valores[valores.length-1-i];
            valores[valores.length-1-i]=v;
        }
    }
}

```

Controlador

```

(en el constructor)
//Añadir escuchadores
v.getGenerar().addActionListener(this::generarNumerosAleatorios);
v.getAscendente().addActionListener(this::ordenarNumerosAscendentemente);
v.getDescendente().addActionListener(this::ordenarNumerosDescendentemente);

//Mover datos a la vista
m.addPropertyChangeListener(Modelo.PROP_VALORES, (
PropertyChangeEvent e)->{
    int i = ((IndexedPropertyChangeEvent)e).getIndex();
    double numero = m.getValores(i);
    numeros[i].setText(String.valueOf(numero));
});

//Acciones
v.getGenerar().doClick();

```

```

(en la clase)
/* METODOS PRIVADOS */
void moverANumeros()
{
    for(int i=0;i<v.lengthNumeros();i++)
    {
        double numero = m.getValores(i);
        v.getNumeros(i).setText(String.valueOf(numero));
    }
}

/* EVENTO generar CLICK */
private void generarNumerosAleatorios(ActionEvent e) {
    m.generar();
    moverANumeros();
}

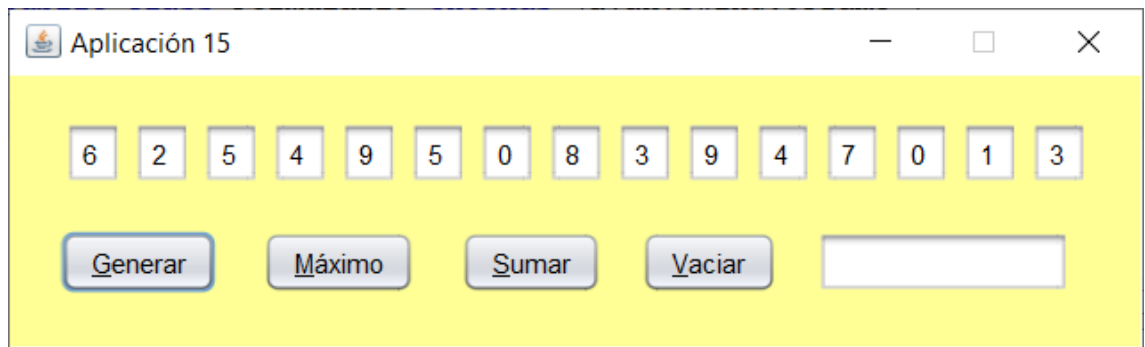
/* EVENTOS ascendente CLICK */
private void ordenarNumerosAscendentemente(ActionEvent e)
{
    m.ordenarAscendentemente();
    moverANumeros();
}

/* EVENTOS descendente CLICK */
private void ordenarNumerosDescendentemente(ActionEvent e)
{
    m.ordenarDescendentemente();
    moverANumeros();
}

```

EJERCICIO

5. Hágase un formulario que permita rellenar 15 cajas de texto con valores aleatorios entre 0 y 9, ambos inclusive, al pulsar un botón. Habrá otros tres botones que permitirán sumar todos los valores, obtener el mayor y vaciar las cajas de texto (asignándoles el valor de “0”). El resultado de la suma y del máximo se almacenará en una caja de texto. Esta caja de texto se vaciará (cadena vacía) al generar o vaciar las 15 cajas de texto.



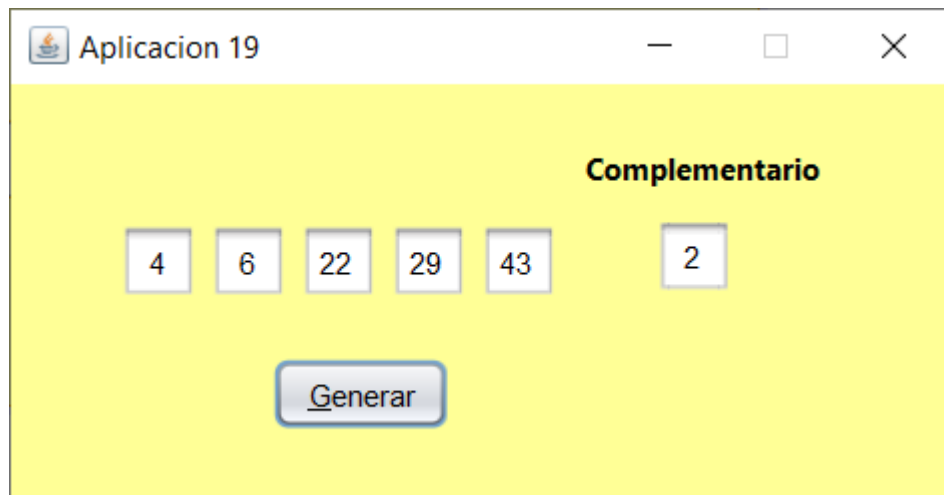
- a) El formulario contendrá un panel de tamaño de 513 x 110 con un relleno de 20 y una separación vertical entre los controles de 20
- b) Todas las cajas de texto son de solo lectura, con un fondo de color blanco y no pueden recibir el foco con el tabulador.
- c) Las primeras 15 cajas de texto tendrán un tamaño de 25 x 25, el texto estará centrado horizontalmente y los controles se separarán entre sí 7.
- d) La última caja de texto tendrá un tamaño de 91 x 25, el texto estará alineado a la derecha y se separa del botón 20
- e) Todos los botones tendrán un tamaño de 75 x 25 y estarán separados entre sí 20

JLabel

PROPIEDADES	
text	Texto actual del control
displayedMnemonic	Tecla de acceso directo
horizontalAlignment	Alineación horizontal
icon	Icono
labelFor	Control de acceso directo
verticalAlignment	Alineación vertical

EJERCICIO

6. Hágase un formulario que muestre el resultado de la Bonoloto (cinco números aleatorios entre 1 y 49, ambos inclusive, y un número aleatorio, llamado complementario, entre 0 y 9, ambos inclusive) al pulsar un botón.



Hágase un diseño lo más parecido al mostrado que mínimamente cumpla las siguientes condiciones:

- Todas las cajas de texto son de solo lectura, con un fondo de color blanco, tienen un tamaño de 30 x 30 y no pueden recibir el foco con el tabulador.
- El texto “Complementario” es el texto de un control JLabel que tiene el siguiente diseño:

font	Segoe UI 12 Bold
text	Complementario

JText (propiedades de validación y eventos)

```
package clases;

import java.io.Serializable;
import java.util.function.Predicate;
import javax.swing.JTextField;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.text.AbstractDocument;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.PlainDocument;

public class JText extends JTextField implements Serializable {
    public static final String PRO_BIND = "bind";
    public static final String PRO_TEXT = "text";

    private Predicate<String> valido = e->true;
    private Predicate<String> completo = e->true;
    private String anterior = "";
    private String cad = getText();

    //private final PropertyChangeSupport pcs = new
    PropertyChangeSupport(this);

    private final DocumentListener dl = new DocumentListener(){
        @Override
        public void insertUpdate(DocumentEvent e) {
```

```

        changedUpdate(e);
    }

    @Override
    public void removeUpdate(DocumentEvent e) {
        changedUpdate(e);
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
        firePropertyChange(PRO_BIND, anterior, cad);
        firePropertyChange(PRO_TEXT, anterior, cad);
    }

};

// <editor-fold defaultstate="collapsed" desc="Métodos get y set">
public Predicate<String> getCompleto() {
    return completo;
}

public void setCompleto(Predicate<String> completo) {
    if(completo==null) this.completo=e->true;
    else this.completo = completo;
}

public Predicate<String> getValido() {
    return valido;
}

public void setValido(Predicate<String> valido) {
    if(valido==null) this.valido=e->true;
    else this.valido = valido;
}
// </editor-fold>

public JText()
{
    this.setDocument(new Documento(this));
    this.getDocument().addDocumentListener(dl);
}

public boolean estaCompleto()
{
    return completo.test(getText());
}

public boolean estaIncompleto()
{
    return !estaCompleto();
}

public class Documento extends PlainDocument {
    private final Object source;

    public Object getSource()
    {
        return source;
    }

    public Documento(Object source)
    {
        this.source=source;
    }

```

```

    }

    @Override
    public void insertString(int offs, String str, AttributeSet a) throws
    BadLocationException
    {
        cad = new StringBuffer(getText(0,
        getLength())).insert(offs, str).toString();

        if(valido.test(cad))
        {
            anterior = getText(0, getLength());
            super.insertString(offs, str, a);
        }
        else super.insertString(offs, "", a);
    }

    @Override
    protected void removeUpdate(AbstractDocument.DefaultDocumentEvent e)
    {
        try
        {
            cad = new StringBuffer(getText(0,
            getLength())).delete(e.getOffset(), e.getOffset()+e.getLength()).toString();

            if(valido.test(cad))
            {
                anterior = getText(0, getLength());
                super.removeUpdate(e);
            }
            else super.insertString(
                e.getOffset(),
                getText(0, getLength()).substring(e.getOffset(),
                e.getOffset()+e.getLength()),
                null);
        }
        catch (BadLocationException ex)
        {
        }
    }
}

```

EVENTOS	
<i>PropertyChangeListener</i>	Lanza el evento cuando se modifica la propiedad “text” del control

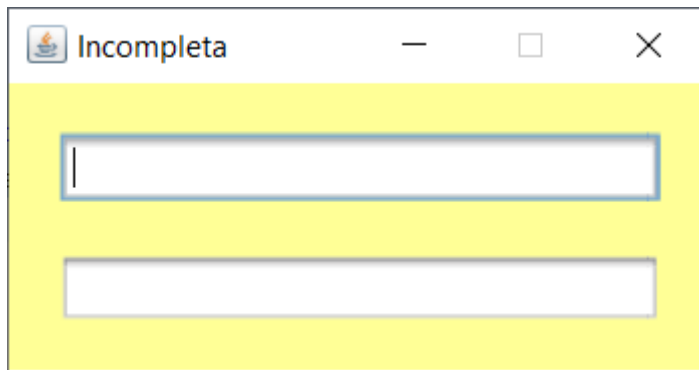
PROPIEDADES	
text	Texto del formulario
completo	Indica si el contenido de la caja de texto está o no completo
valido	Indica si el contenido de la caja de texto es o no válido (la cadena vacía siempre es válida)

EJERCICIO (sólo el diseño)

- Hágase un formulario que tenga dos cajas de texto de forma que la segunda caja de texto sea de solo lectura y en la que se haga lo siguiente:

- a) A medida que se escribe en la primera caja de texto, se copia dicho contenido en la segunda caja de texto.
- b) En la primera caja de texto sólo se podrá escribir palabras en mayúsculas con acentos y diéresis que tengan como máximo 10 caracteres.
- c) El contenido de la primera caja de texto estará completo si tiene al menos 5 letras.
- d) También se pondrá la misma cadena en minúsculas en minúsculas como título del formulario siempre y cuando esté completa y si no está completa, se mostrará en el título que está incompleta.

(Aplicación 17)



- a) El formulario contiene un panel “root” de tamaño 280 x 104, centrado, no redimensionable y con el fondo de color 255, 255, 150 con un relleno de 20
- b) Dos cajas de texto. “origen” y “destino”, de ancho 240. La segunda es de solo lectura con fondo blanco

Modelo

```
public static boolean palabra(String t) {
    if(t.matches("[A-ZÑÁÉÍÓÚ]{0,10}")) return true;

    return false;
}

public static boolean palabraCompleta(String t) {
    if(t.matches("[A-ZÑÁÉÍÓÚ]{5,}")) return true;

    return false;
}

public static void invertirMinusculas(String palabra)
{
    String s = new
StringBuffer(palabra).reverse().toString().toLowerCase
();
}
```

```

        return s;
    }

```

Controlador

```

(en el constructor)
//Añadir escuchadores
v.getOrigen().addPropertyChangeListener(JText.PRO_TEXT
, this::obtenerDestinoYTitulo);

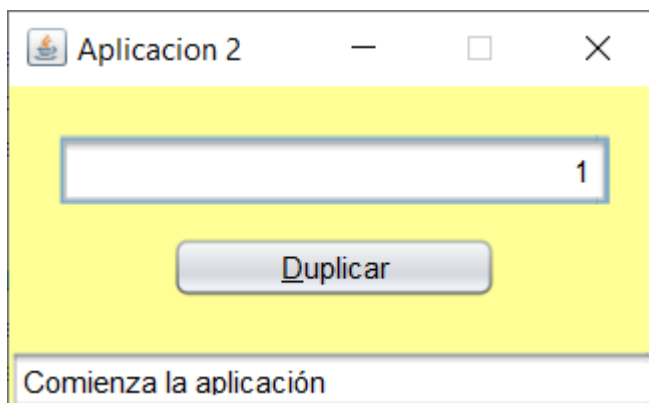
/* EVENTO origen.text CHANGED */
private void obtenerDestinoYTitulo(PropertyChangeEvent
evt)
{
    String palabra = v.getOrigen().getText();
    v.getDestino().setText(palabra);
    if(!v.getOrigen().estaCompleto())
    {
        v.setTitle("Incompleta");
        return;
    }

    palabra = Modelo.invertirMinusculas(palabra);
    v.setTitle(palabra);
}

```

EJERCICIO

8. Hágase un formulario que permita duplicar el valor de una caja de texto que inicialmente tiene el valor de 1.



- a) El panel tendrá un tamaño de 260 x 130
- b) La caja de texto tendrá un tamaño de 220 x defecto y un margen superior, derecho e izquierdo de 20.
- c) El botón tendrá un tamaño de 130 x 25, un margen superior de 12, margen izquierdo y derecho de 65 y margen inferior de 20

- d) La caja de texto de estado tendrá un tamaño de 260 x 25 y será de solo lectura.
- e) En la caja de texto sólo se podrá introducir valores entre 0 y 99999, ambos inclusive, o la cadena vacía.
- f) El botón estará habilitado mientras el contenido de la caja de texto sea distinto de la cadena vacía.
- g) Al modificarse el contenido de la caja de estado se pondrá en la barra de estado “Valor modificado” y al pulsar sobre el botón se pondrá “Valor duplicado”
- h) Al pulsar el botón, se duplicará el valor actual de la caja de texto si su doble no supera el valor de 99999. De superarlo, se actualizará a uno más que al iniciarse el proceso de duplicado. Si el nuevo valor de inicio fuera superior a 49999, se establecerá a 1.

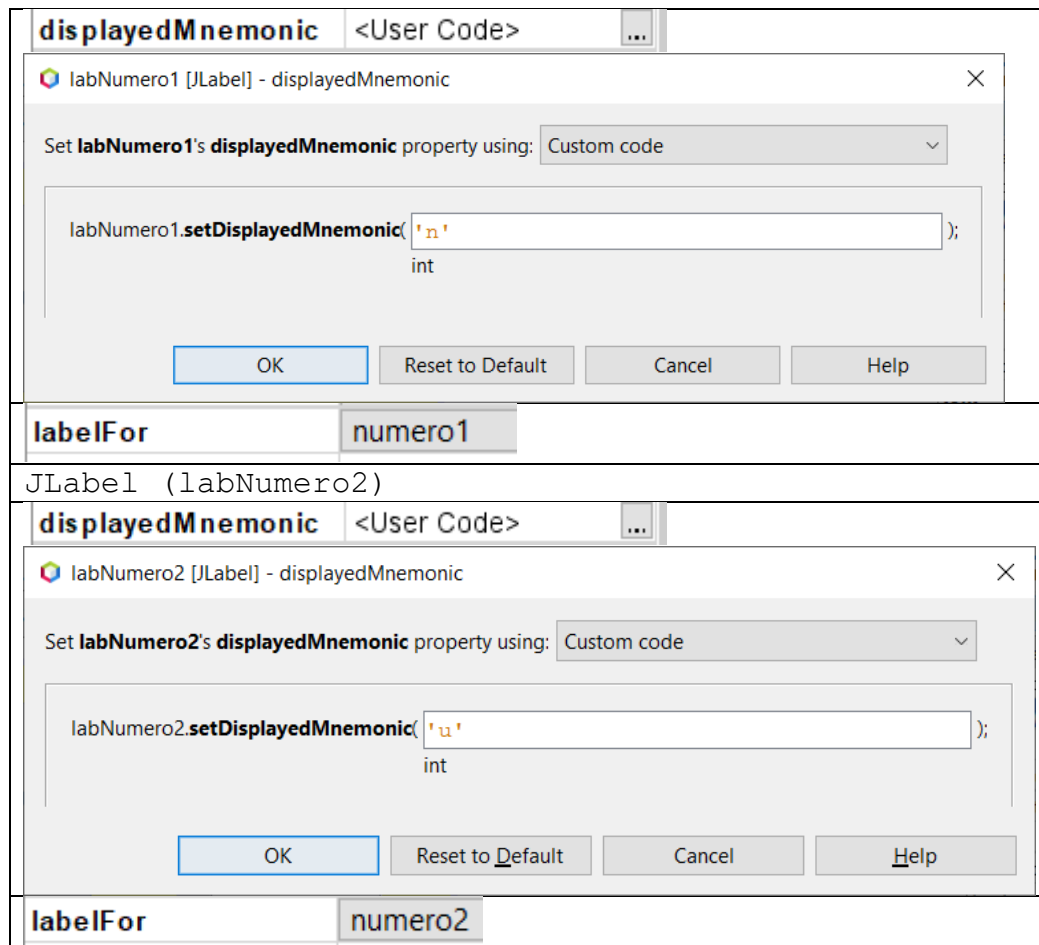
Asociar un mismo escuchador y uso de clases envoltorio

9. Hágase un formulario que permita sumar dos enteros que toman valores entre -1000 y 1000, ambos incluidos.

- a) El panel principal tiene un tamaño de 250 x 155 con un relleno horizontal de 40 y vertical de 20. Las cajas de texto tienen un tamaño de 100 x 25 y las etiquetas de 50 x 25. Cada caja de texto se separa de su etiqueta 20 y entre las etiquetas y cajas de texto hay una separación de 20.
- b) Los datos se introducen en sendas cajas de texto de entrada que sólo admiten los valores enteros entre -1000 y 1000, ambos inclusive, y la cadena vacía.
- c) El resultado de la suma se pone en una caja de texto de solo lectura que no puede recibir el foco con el tabulador.
- d) El cálculo de la suma se hace a medida que se escriben los valores en las cajas de texto de entrada. Si el valor de alguna de tales cajas de texto fuera la cadena vacía, el resultado de la suma será la cadena vacía.
- e) La caja de texto de entrada que tenga el foco se pondrá con fondo de color amarillo.
- f) Cuando las cajas de entrada reciben el foco se ponen con fondo amarillo y al perderlo se ponen con fondo blanco.

Poner teclas de acceso directos en etiquetas a controles

```
JLabel (labNumero1)
```



Clases envoltorio

Modelo

```
private Integer resultado = 0;
```

Asignar un mismo método de validación y completo a distintas cajas de texto

Las cajas de entrada se denominan “numero1” y “numero2”

```
//Modificar la vista
v.getNumero1().setValido(Modelo::introducirEntero);
v.getNumero1().setCompleto(Modelo::formatoEntero);

v.getNumero2().setValido(Modelo::introducirEntero);
v.getNumero2().setCompleto(Modelo::formatoEntero);
```

Asignar un mismo escuchador a distintas cajas de texto

Controlador

```
(en el constructor)
//Añadir escuchadores
```



```

numero1.addPropertyChangeListener(JText.PRO_TEXT,
vNumerosTextChanged);
numero2.addPropertyChangeListener(JText.PRO_TEXT,
vNumerosTextChanged);

```

```

numero1.addFocusListener(vIntercambiarFondo);
numero2.addFocusListener(vIntercambiarFondo);

```

Obtener la caja de texto que lanzó el evento en los eventos de “intercambiarFondo”

Controlador

(en la clase)

```

/* EVENTO numero1,numero FOCUS */
//private class intercambiarFondo implements FocusListener
private class intercambiarFondo extends FocusAdapter {

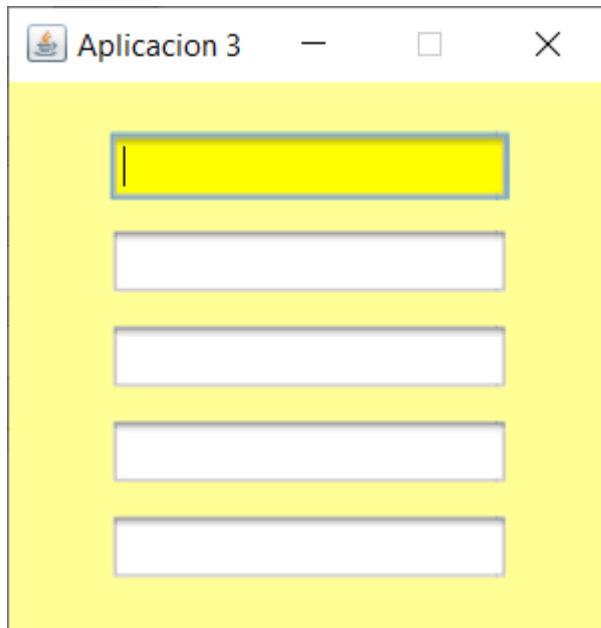
    @Override
    public void focusLost(FocusEvent e) {
        JTextField caja = (JTextField) e.getSource();
        caja.setBackground(Color.white);
    }

    @Override
    public void focusGained(FocusEvent e) {
        JTextField caja = (JTextField) e.getSource();
        caja.setBackground(Color.yellow);
    }
}

```

EJERCICIO

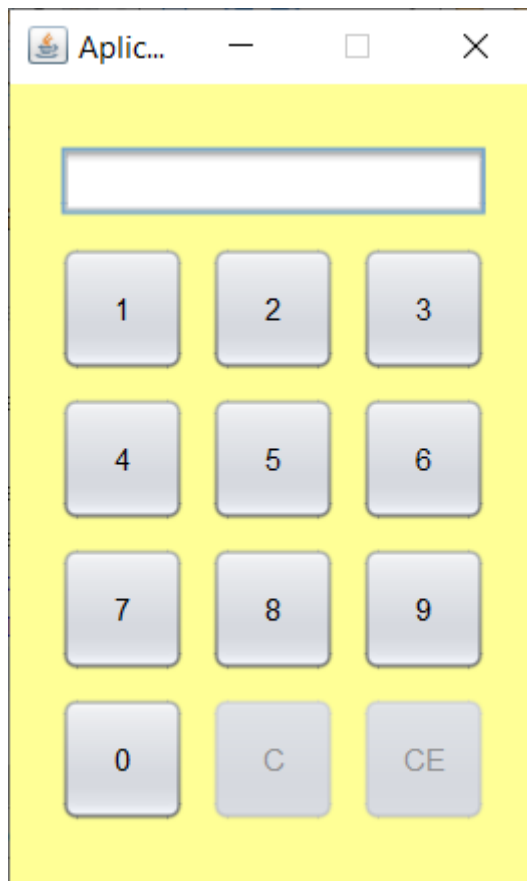
10. Hágase un formulario que tenga 5 cajas de texto (4 de entrada y 1 de salida). Cada una de las cajas de texto de entrada se pondrán con fondo blanco cuando tomen el foco y con fondo blanco cuando lo pierda.



- a) El panel principal tiene un tamaño de 240 x 230 con un relleno vertical de 20 y horizontal de 40
- b) Todas las cajas de texto tienen una anchura de 160 y una altura de 30 y están separadas entre sí 10 y pueden recibir el foco con el tabulador salvo la última.
- c) En las cajas de entrada sólo se pueden introducir palabras en mayúsculas sin acentos ni diéresis.
- d) Al escribir en cualquiera de las cajas de entrada se copiará su contenido en la caja de salida.

EJERCICIO

11. Hágase el siguiente formulario (Aplicación 32)

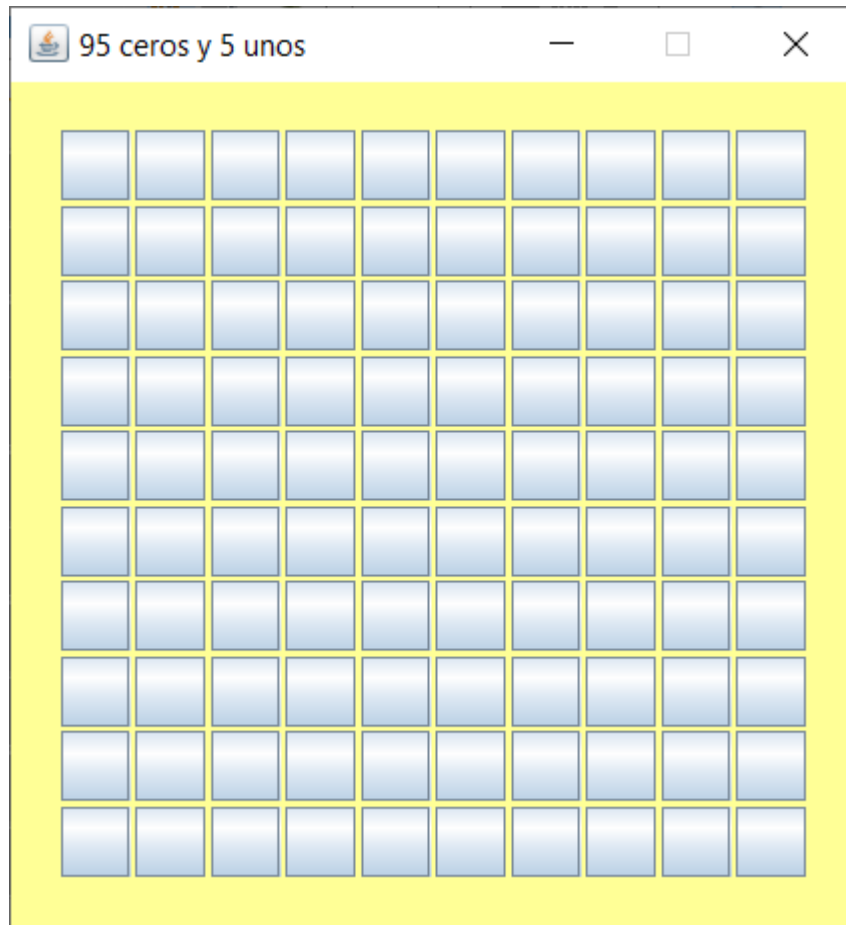


- a) El panel principal tiene un tamaño de 210 x 305 con un relleno de 20 y el texto alineado hacia la derecha
 - b) La caja de texto es de solo lectura de 170 x 25 y separada de los botones 10
 - c) Todos los botones tienen un tamaño de 50 x 50 y separados entre sí horizontal y verticalmente 10.
- 1) Al pulsar sobre los botones “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9” se pondrá el texto del botón en la caja de texto si estuviera vacía y se añadirá dicho texto por la derecha si tuviera contenido.
 - 2) Al pulsar el botón “C” se quitará el último dígito por la derecha del contenido de la caja de texto.
 - 3) Al pulsar el botón “CE” se vaciará la caja de texto.
 - 4) Los botones “C” y “CE” estarán deshabilitados si no hay nada como contenido de la caja de texto
 - 5) Los botones “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9” estarán deshabilitados si la caja de texto tiene un número con 7 dígitos.
 - 6) Los botones “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9” estarán deshabilitados si el contenido de la caja de texto es “0”

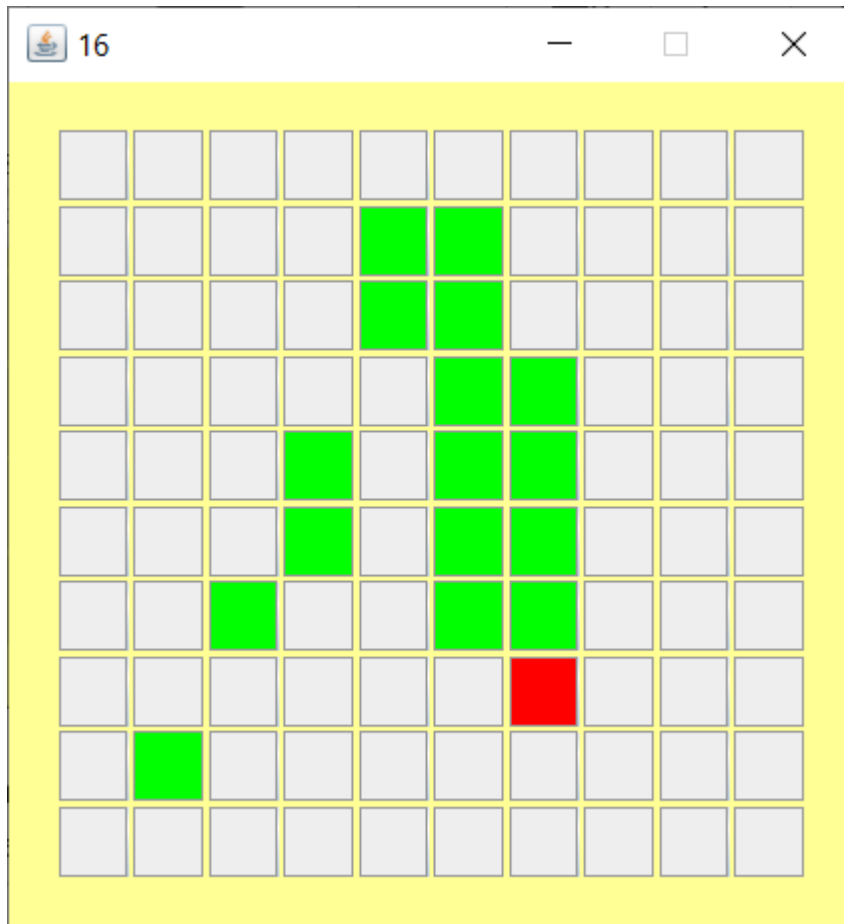
Generar el código de la vista y tabla de controles

12. Considérese el siguiente juego: Se genera un tablero de 10x10 con 95 celdas con el valor de 0 y 5 celdas con el valor de 1 de forma aleatoria. Se van descubriendo

celdas hasta que se descubre una celda con el valor de 1. Se muestra en el título del formulario el número de celdas descubiertas con el valor de 0. (Aplicación 30)



- a) Formulario con una apariencia y sensación de “Metal” y el panel principal tiene un tamaño de 338 x 338
 - b) Los botones tienen un tamaño de 28 x 28 y no pueden recibir el foco con el tabulador. Hay una separación horizontal y vertical entre los botones de 2
 - c) El panel tiene un relleno de 20
-
- 1) Al pulsar sobre un botón se pone de color verde si contiene un 0 y rojo si contiene un 1, y se deshabilita dicho botón.
 - 2) Al descubrir el primer 1, se muestra un mensaje en el título del formulario con el número de celdas con el valor 1 descubiertas, y se deshabilitan todos los botones.



Generar los botones en el código y definir una tabla de controles

Vista

(en la clases)

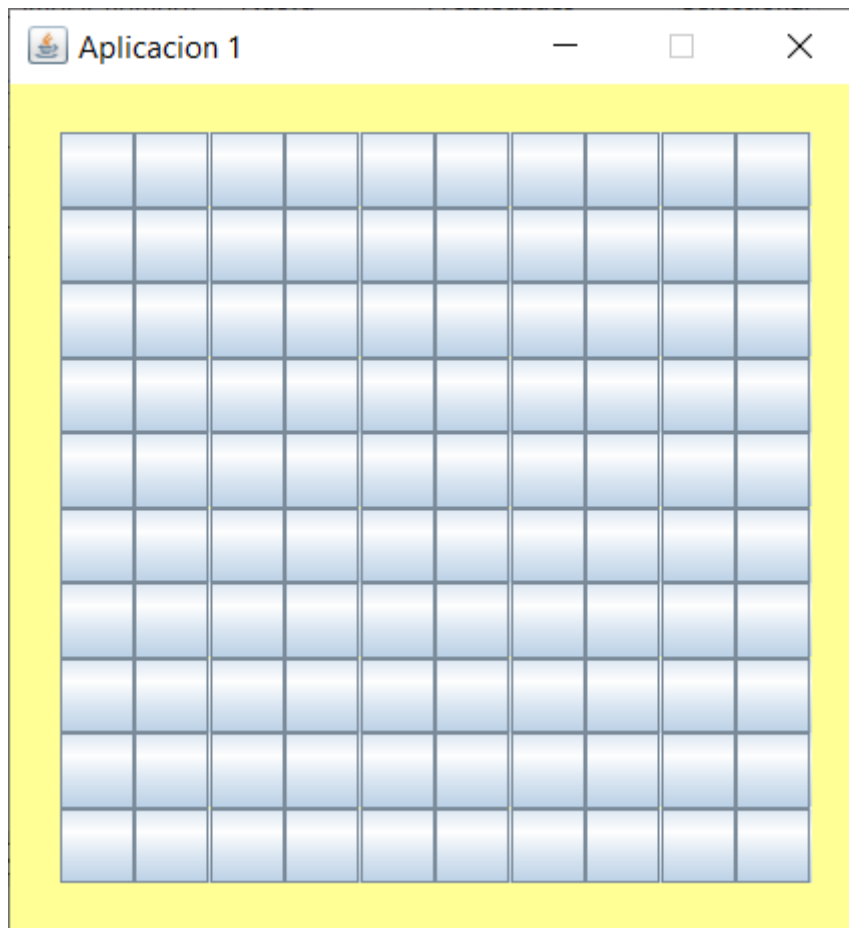
```
private JButton[][] botones;
```

(en el constructor)

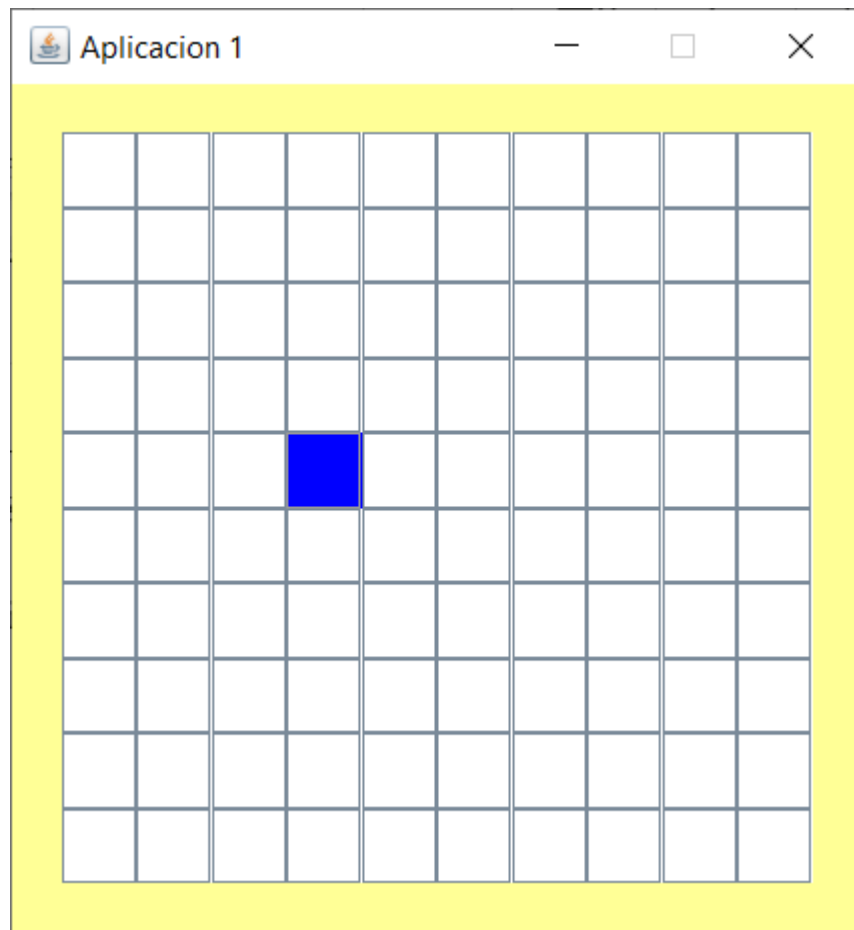
```
root.setLayout(null);
botones = new JButton[10][10];
for(int i=0;i<botones.length;i++)
    for(int j=0;j<botones[i].length;j++)
    {
        botones[i][j] = new JButton();
        root.add(botones[i][j]);
        botones[i][j].setBounds(30*i+20,30*j+20, 28, 28);
    }
```

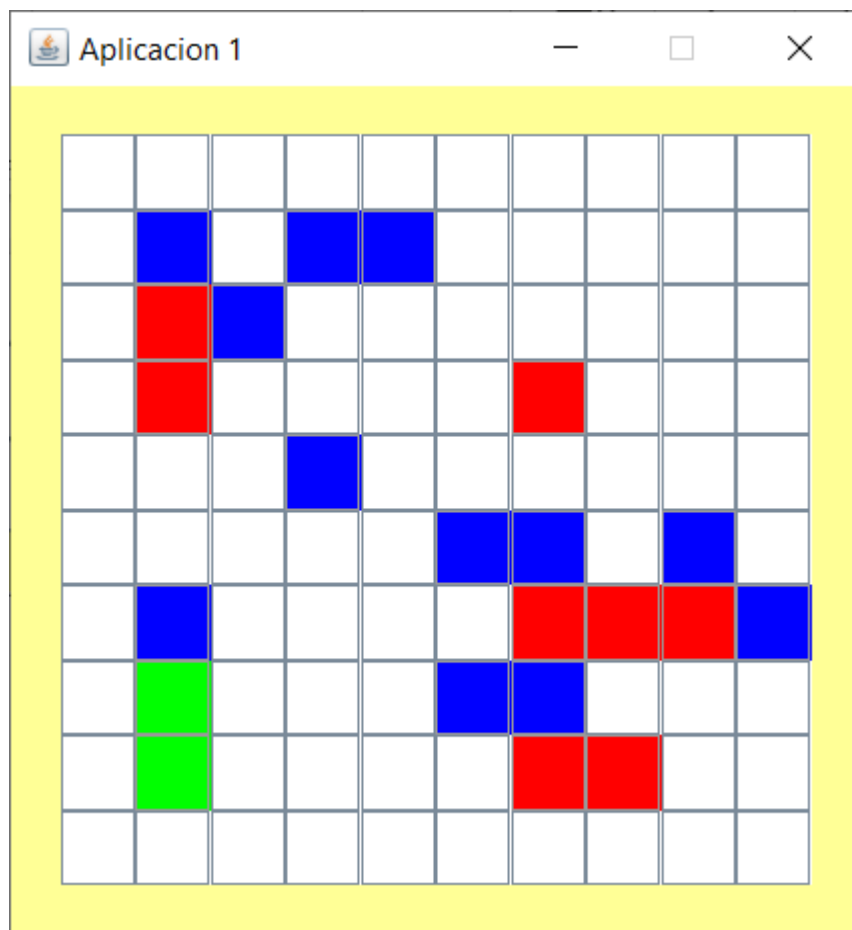
EJERCICIO

- 13.** Prográmese el juego de los barquitos. El juego consiste en generar 10 barcos de las siguientes características: 1 barco de tamaño 4; 2, de tamaño 3; 3, de tamaño 2 y 4, de tamaño 1. Los barcos pueden disponerse en el tablero de 10 x 10 de forma horizontal o vertical, pero no se podrán tocar.

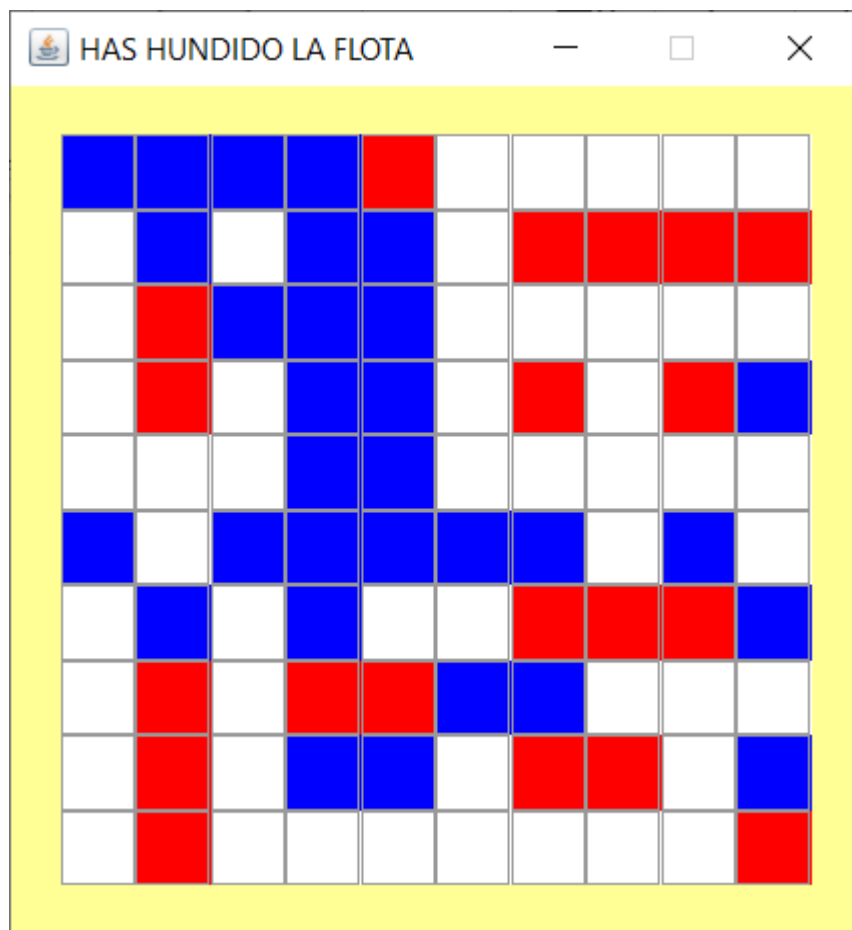


- a) Formulario con una apariencia y sensación de “Metal” y el panel principal tiene un tamaño de 330 x 330 con un relleno de 20
 - b) Los controles se disponen en el panel principal de forma absoluta.
 - c) Los botones tienen un tamaño de 30 x 30 y no pueden recibir el foco con el tabulador.
-
- 1) Al pulsar sobre un botón se comprueba si se ha pulsado sobre agua, poniéndose el botón con color azul; si se ha tocado un barco, poniéndose en color verde o si se ha hundido, poniéndose todos los botones del barco hundido en rojo. El botón se deshabilita.



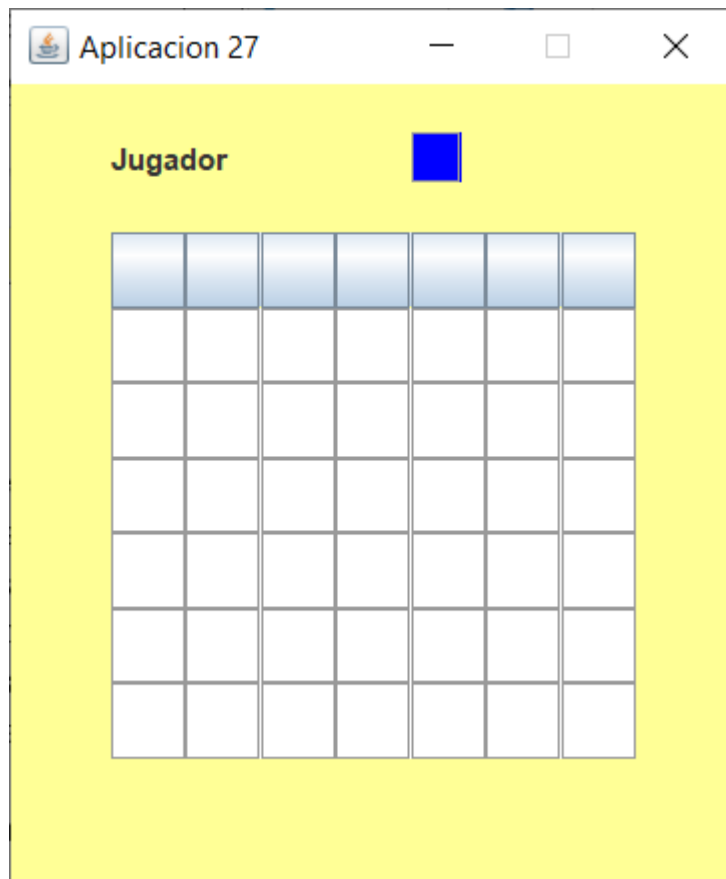


- 2) Si se hunden todos los barcos, se deshabilitan todos los botones y se pone un mensaje en el título del formulario indicando que se ha hundido la flota.



EJERCICIO

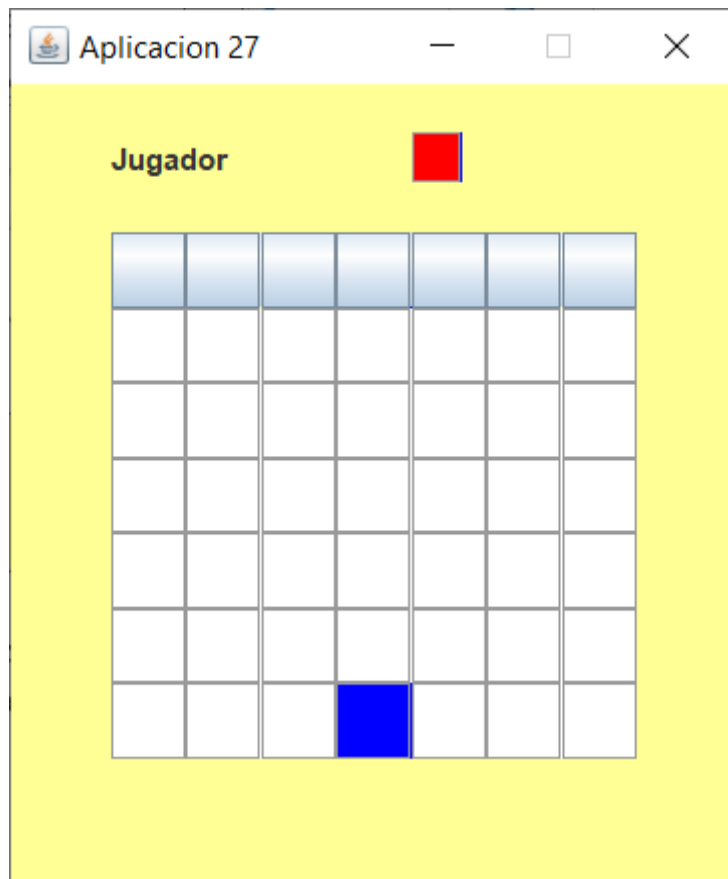
- 14.** Prográmese el juego de las cuatro en raya. El tablero tendrá un tamaño de 6 filas y 7 columnas. De forma alternativa los jugadores irán colocando fichas (fondo azul y rojo) hasta que consigan 4 en raya de forma horizontal, vertical o diagonal. El juego termina cuando un jugador hace 4 en raya en su último movimiento o bien se rellena el tablero y ningún jugador ha hecho 4 en raya, en este último caso se dice que han empatado.



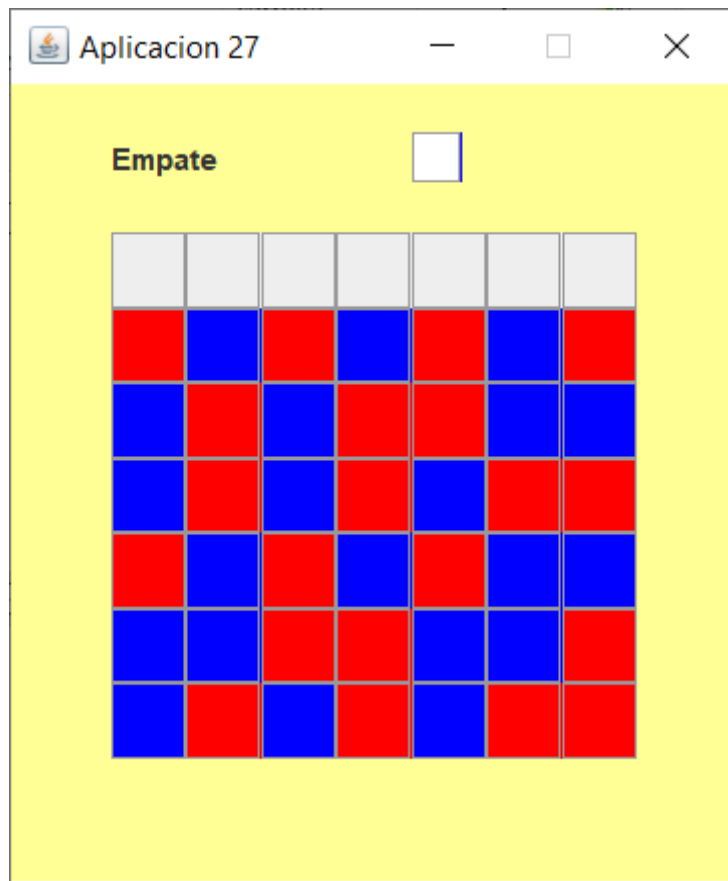
- a) Formulario con una apariencia y sensación de “Metal” y el panel principal tiene un tamaño de 290 x 320 con un relleno superior de 20, inferior de 60 y derecho e izquierda de 20
 - b) Los controles se disponen en el panel principal de forma absoluta.
 - c) La etiqueta tiene un tamaño de 100 x 20 y está a una distancia horizontal de 40 y vertical de 20
 - d) El botón donde se indica el jugador que le toca jugar actualmente, tiene un tamaño de 20 x 20, está a una distancia horizontal de 160 y vertical de 20, está deshabilitado y tiene un fondo de color azul (primer jugador que juega).
 - e) Todos los botones de la tabla de botones tienen un tamaño de 30 x 30. La tabla está a una distancia horizontal de 40 y vertical de 60. Salvo los botones de la primera fila, el resto están deshabilitados y tienen un fondo de color blanco.
- 1) Al pulsar sobre uno de los botones de la primera fila, se colorea el primer botón que haya libre desde la parte inferior con el color del jugador correspondiente. Una vez coloreado, se comprueba si el jugador ha hecho 4 en raya. De hacer 4 en raya, se deshabilitan todos los botones y se pone en la etiqueta el texto de “Ganador” y en el botón el color del jugador ganador.



De no hacer 4 en raya, se cambia de turno del jugador indicándose en el botón el color del nuevo jugador.



Puede ser que el juego termine en empate al completarse todas las casillas y ningún jugador haya obtenido 4 en raya. En tal caso se muestra en la etiqueta el texto de “Empate” y el botón se pone con fondo blanco.



CONTROLES BÁSICOS DE FORMULARIOS

- Controles de barras horizontales y verticales → JScrollBar
- Controles deslizadores horizontales y verticales → JSlider
- Controles de botones de opción y grupo de botones → JRadioButton y ButtonGroup
- Control de caja de texto multilíneas → JTextArea
- Control de casilla de verificación → JCheckBox
- Componente de temporizador → Timer
- Control de tipo botón gráfico booleano → JToggleButton

JScrollBar

EVENTOS

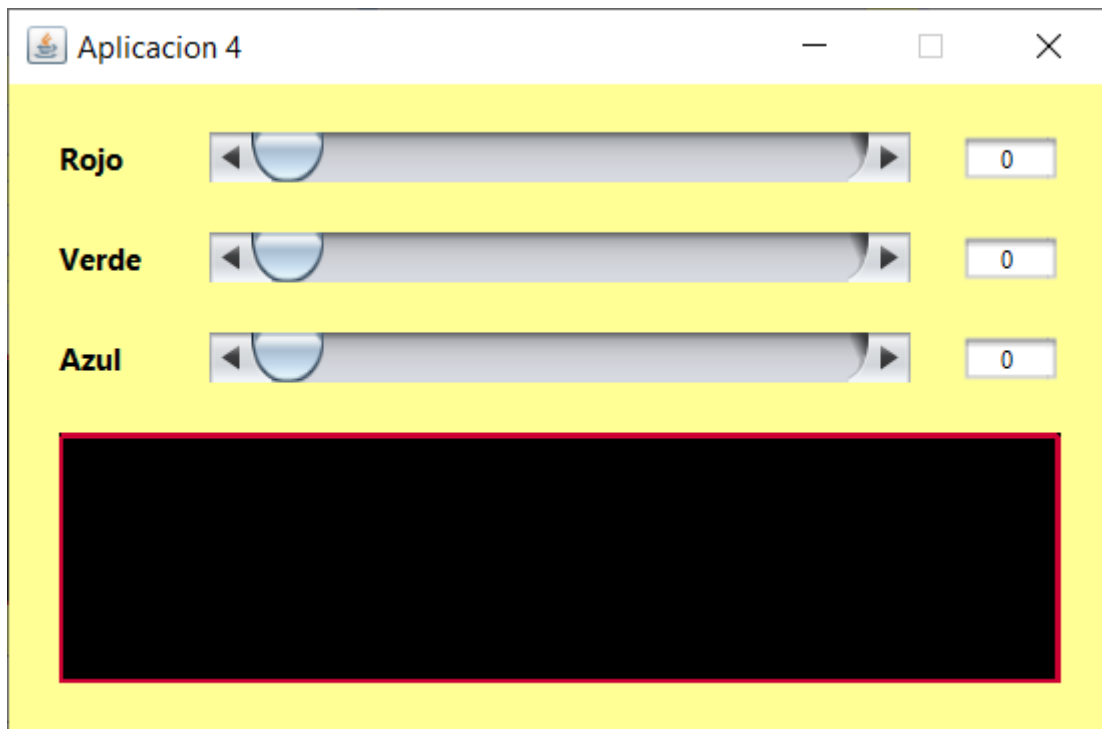
<i>AdjustmentValueChanged</i>	Se lanza el evento cuando se cambia el valor de la propiedad “value”
-------------------------------	--

PROPIEDADES

value	Valor actual del control
orientation	Orientación del control
máximo	Máximo valor
mínimo	Mínimo valor
blockIncrement	Incremento al pulsar sobre la barra
unitIncrement	Incremento al pulsar sobre los extremos de la barra (flechas)
visibleAmount	Cantidad que ocupa el botón de desplazamiento (si vale 10 y queremos que tome valores entre 0 y 100 hay que establecer como valor máximo el de 110)

EJERCICIO

15. Hágase un formulario que permita establecer el color de un panel con los matices de los colores de rojo, verde y azul introducidos en tres barras horizontales. Los matices de los colores rojo, verde y azul se mostrarán en sendas cajas de texto de solo lectura.



- El panel tiene un tamaño de 440 x 260 y un borde “Line Border” con un color 204, 0, 51 y un relleno de 20.
- Todos los controles se separan 20 entre sí horizontal y verticalmente.
- Las etiquetas tendrán un tamaño de 40 x 20 y un tipo de fuente “Segoe UI 12 Bold”
- Las barras horizontales tendrán un tamaño de 280 x 20 y toman valores entre 0 y 265.
- Las cajas de texto tienen un tamaño de 40 x 20, son de solo lectura con el texto centrado y la letra tienen un tamaño de letra de 10

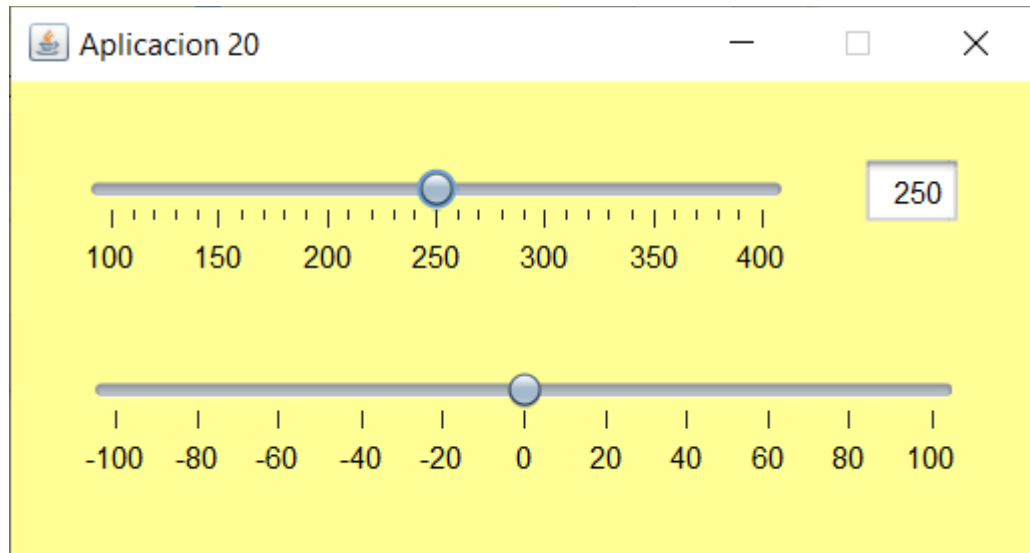
JSlider

EVENTOS	
<i>ChangeListener</i>	Se lanza el evento cuando se cambia el valor de la propiedad “value”

PROPIEDADES	
value	Valor actual del control
orientation	Orientación del control
maximum	Máximo valor
mínimum	Mínimo valor
majorTickSpaing	Cada cuánto se muestran los marcadores principales. El primer valor que se ponga sirve para establecer cada cuanto se muestran los valores a partir del primero.
minorTickSpacing	Cada cuánto se muestran los marcadores secundarios
paintLabels	Mostrar los valores
paintTicks	Mostrar los marcadores
paintTrack	Mostrar la barra
snapToTicks	

EJERCICIO

16. Hágase un formulario que sume los valores de dos barras de deslizamiento nada más modificarse tales controles.



- f. El panel tiene un tamaño de 410 x 190 y tiene un relleno de 30ç
- g. Los controles deslizadores tienen un tamaño de 280 x 50 y 350 x 50, respectivamente.
- h. La caja de texto tiene un tamaño de 40 x defecto, es de solo lectura y el texto está alineado a la derecha.

JRadioButton y ButtonGroup

JRadioButton

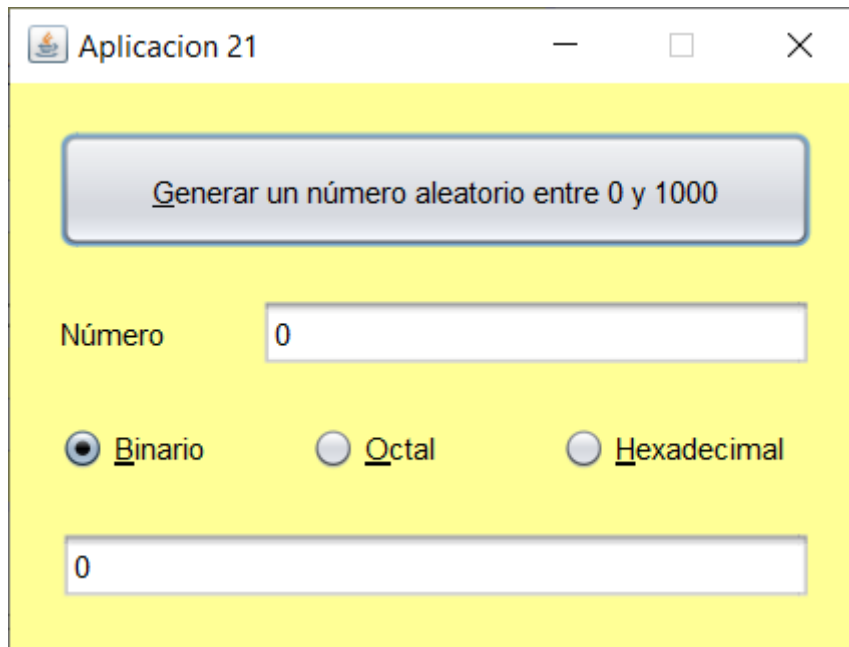
EVENTOS	
<i>ItemListener</i>	Se lanza el evento cuando se cambia el valor de la propiedad “selected”
<i>ActionListener</i>	Se lanza el evento cuando se pulsa sobre el control. Se ejecuta después de ItemListener

PROPIEDADES	
selected	Si está o no activado
buttongroup	Control de agrupamiento
mnemonic	Tecla de acceso directo
text	Texto actual del control

EJERCICIO

17. Hágase un formulario que permita generar números aleatorios entre 0 y 1000, ambos inclusive, y convertirlos a binario, octal o hexadecimal.

- a) Al pulsar sobre un botón se genera el número aleatorio, que se almacena en una caja de solo lectura con fondo blanco, y se convierte a la base que esté activada. El número convertido se muestra en una caja de texto de sólo lectura.
- b) Al activar un botón de opción, se convierte el número aleatorio a la nueva base.



- i. El panel tiene un tamaño de 340 x 215
- j. El botón tiene un tamaño de 300 x 46
- k. La etiqueta tiene un tamaño de 60 x defecto
- l. La primera caja de texto tiene un tamaño de 220 x 20
- m. Los dos primeros botones de opción tienen un tamaño de 80 x 25
- n. El tercer botón de opción tiene un tamaño de 100 x 25
- o. La segunda caja de texto tiene un tamaño de 300 x defecto
- p. Todos los controles se separan entre sí horizontal y verticalmente 20

JTextArea y JCheckBox

JTextArea (de sólo lectura)

PROPIEDADES	
text	Texto actual del control
editable	De lectura o escritura
columns	Número de columnas preferibles a mostrar
margin	Margen superior, derecho, inferior, izquierdo
lineWrapp	Texto enrollable o no
rows	Número de filas preferibles a mostrar
tabSize	Número de caracteres a expandir
wrapStyleWord	Estilo de las palabras al enrollarse

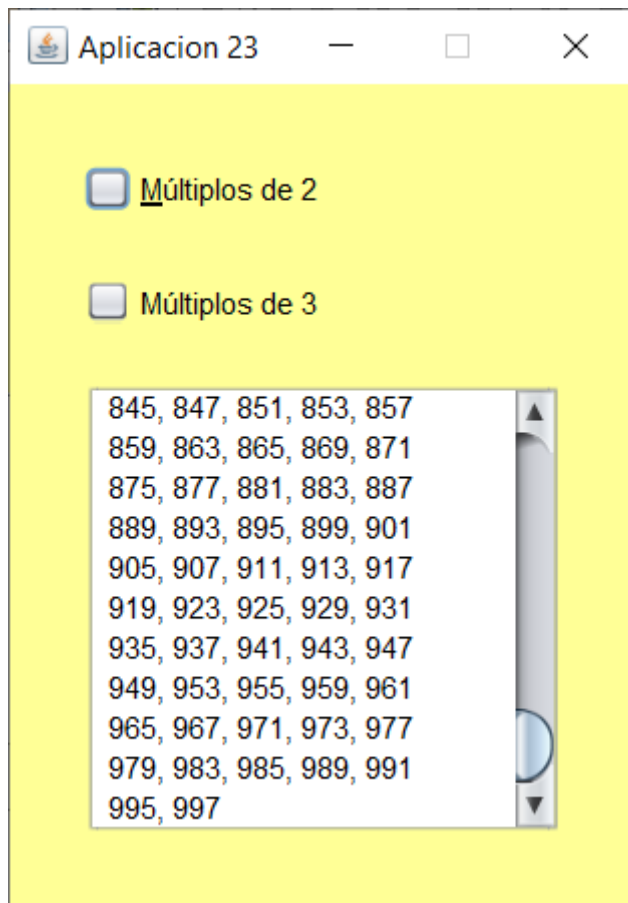
JCheckBox

EVENTOS	
<i>ItemListener</i>	Se lanza el evento cuando se cambia el valor de la propiedad "selected"
<i>ActionListener</i>	Se lanza el evento cuando se pulsa sobre el control. Se ejecuta después de ItemListener

PROPIEDADES	
selected	Si está o no activado
buttongroup	Control de agrupamiento
mnemonic	Tecla de acceso directo
text	Texto actual del control

EJERCICIO

- 18.** Hágase un formulario que tiene dos casillas de verificación con los textos "Múltiplo de 2" y "Múltiplo de 3" sin estar seleccionadas. También hay una caja de texto multilínea de solo lectura y fondo blanco. Inicialmente en la caja de texto están todos los números entre 1 y 1000 que no son múltiplos de 2 ni de 3. En dicho control se mostrarán entre los números del 1 al 1000 los múltiplos de 2 si sólo esta activada la primera casilla; los múltiplos de 3, si sólo está activada la segunda casilla; los múltiplos de 2 y de 3, si están activadas las dos casillas, y los que no son múltiplos de 2 ni de 3, si están desactivadas las dos casillas. En cuanto se modifique el estado de alguna de las casillas, se actualizará la caja de texto.



- q. El panel tiene un tamaño de 250 x 330 y con un relleno de 30
- r. Las casillas de verificación tienen un tamaño de 120 x 25
- s. El panel que contiene a la caja de texto multilínea tiene un tamaño de 190 x 180
- t. Los controles están separados verticalmente 20

Temporizadores (Javax.swing.Timer)

EVENTOS	
<i>ActionListener</i>	Se lanza el evento cada cierto tiempo

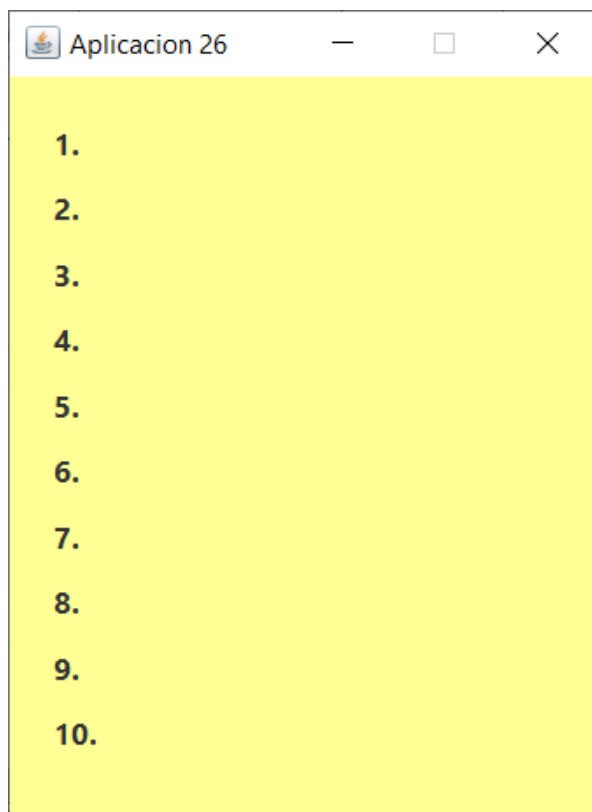
METODOS	
Timer	Constructor que establece el retardo en milisegundos inicial y entre disparos, y establece el primer evento (se puede establecer a null). El retardo inicial y entre disparos y los eventos se pueden modificar independientemente después.
set y add	
setDelay	Establece el retraso entre eventos, el número de milisegundos entre eventos de acción sucesivos.
setInitialDelay	Establece el retraso inicial, el tiempo en milisegundos a esperar después de que se inicia el temporizador antes de disparar el primer evento.

addActionListener	Añade un evento al disparador
get	
getInitialDelay	Devuelve el retraso inicial
getDelay	Devuelve el retraso, en milisegundos, entre disparos de eventos de acción
getActionListener	Devuelve todos los eventos registrados en el temporizador
consulta	
isRunning	Devuelve verdadero si es temporizador se esta ejecutando
Cambio de estado	
restart	Reinicia el Timer, cancelando cualquier disparo pendiente y haciendo que se dispare con su retraso inicial.
start	Inicia el temporizador, lo que hace que comience a enviar eventos de acción a sus oyentes.
stop	Detiene el temporizador, lo que hace que deje de enviar eventos de acción a sus oyentes.
removeActionListener	Elimina un evento registrado

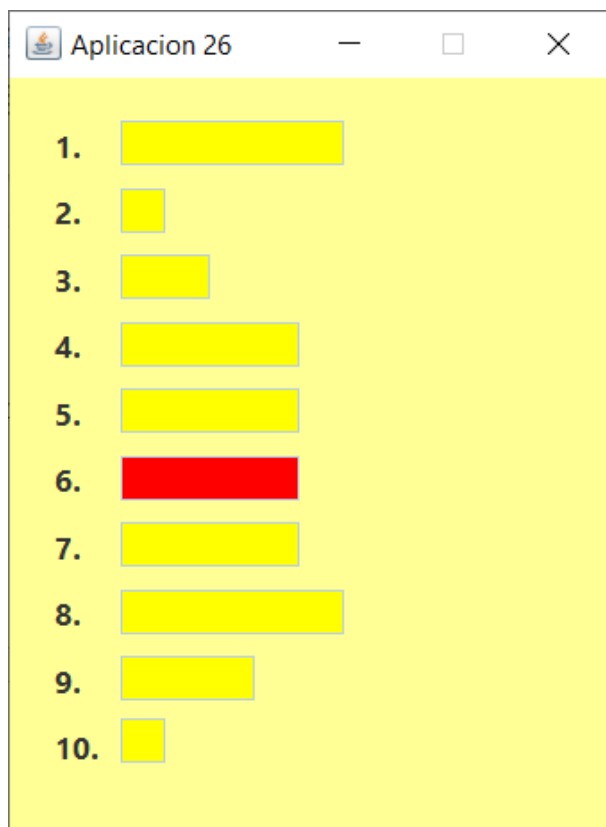
Los temporizadores se crean en la vista con un tiempo de retardo (inicial y entre disparos) de 0 y con un evento null. En el controlador se llaman a los métodos addActionListener, setDelay y setInitialDelay para establecer el evento, el retardo entre disparadores y el retardo inicial.

EJERCICIO

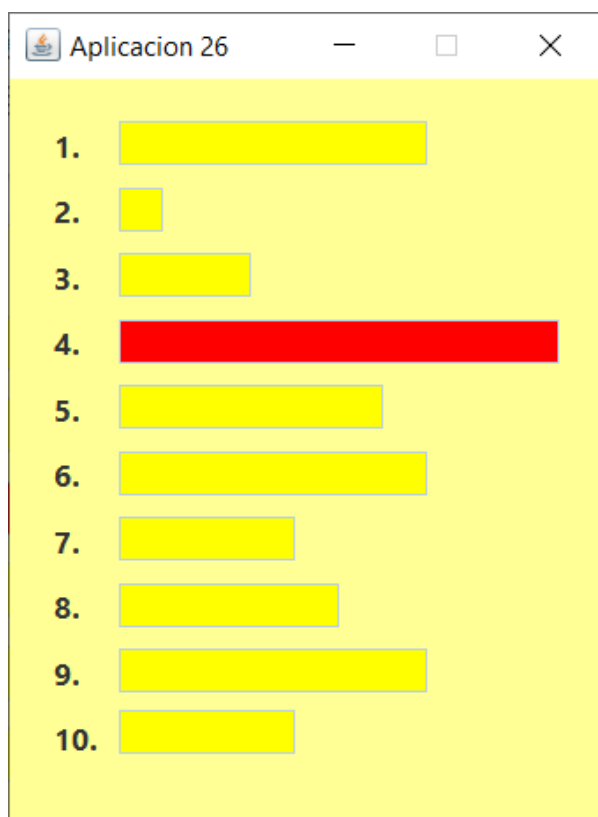
19. Simúlese el juego de la carrera de caballos.



carga inicial



carga intermedia



carga final

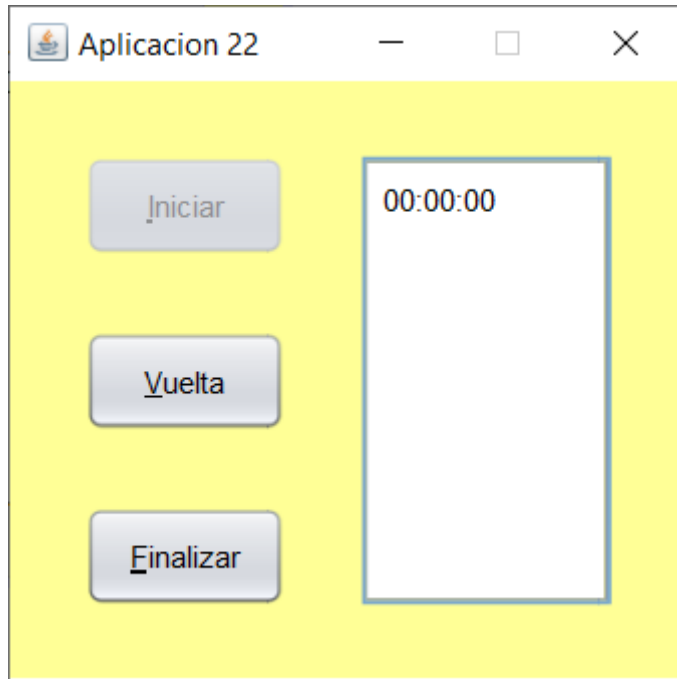
- a) El formulario tendrá una apariencia y sensación de “Metal”.
 - b) El panel principal tendrá un tamaño de 270 x 340 con un margen de 20
 - c) Las 10 etiquetas tendrán un tamaño de 20 x 20
 - d) Las 10 cajas de texto serán de sólo lectura, tendrá un fondo de color amarillo y un tamaño de 0 x 20 (0 de ancho).
 - e) Las etiquetas y cajas de texto se separarán entre sí horizontal y verticalmente 10
- 1) Habrá un temporizador que se lanza dos segundos después de cargarse el formulario y cuyo evento se ejecuta cada segundo.
 - 2) Cada vez que pase 1 segundo, se incrementará en 20 el ancho de uno de los controles de caja de texto de forma aleatoria, y se pondrá como color de fondo de dicho control el de rojo. Si dicho control tuviera un ancho de 200, entonces se para el temporizador.
 - 3) El formulario no se podrá cerrar hasta que termine la carrera.

EJERCICIO

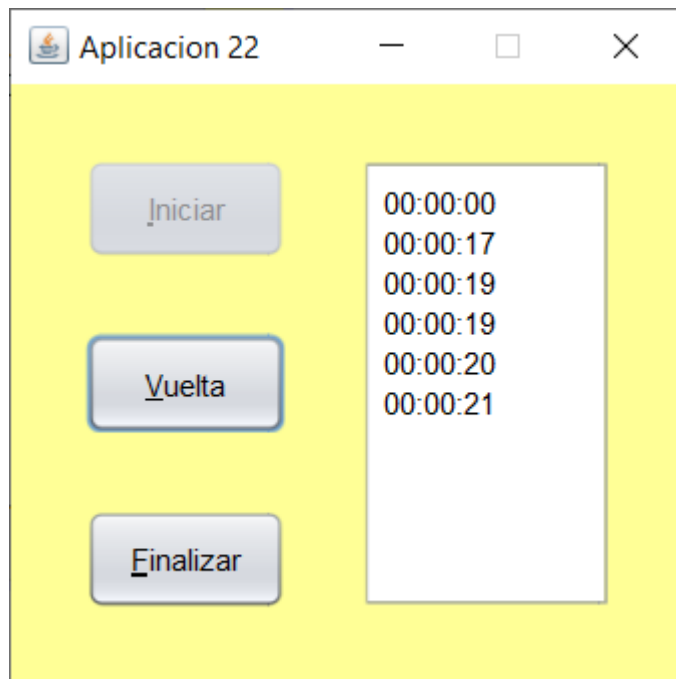
20. Hágase el siguiente formulario

- a) El panel principal tendrá un tamaño de
- b) Los botones tienen un tamaño de 80 x 40
- c) El panel contenedor de la caja de texto multilínea tiene un tamaño de 100 x 180

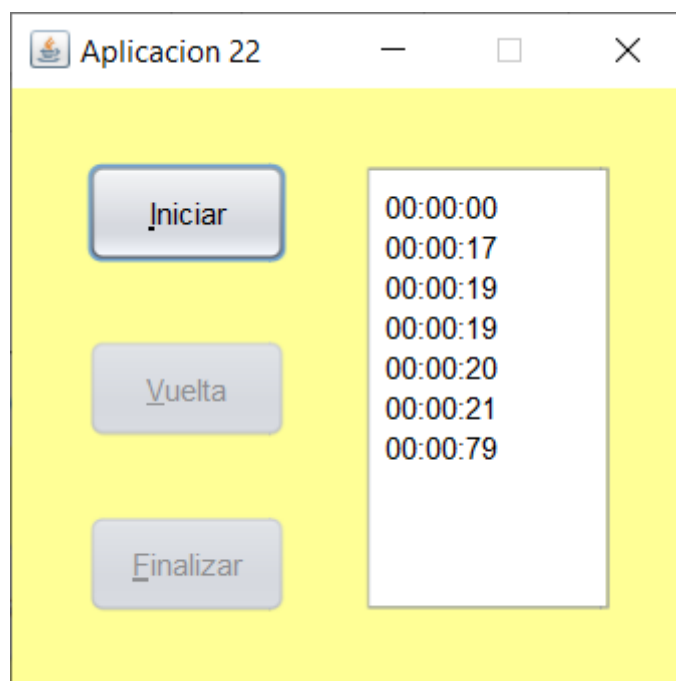
- d) Los botones y el panel contendor de los tiempos se separan horizontal y verticalmente entre sí 30
 - e) La caja de texto multilínea es de sólo lectura tiene 10 columnas y 5 filas para mostrar de forma preferible.
- 1) Al pulsar el botón “Iniciar” se habilita los botones “Vuelta” y “Finalizar”, y se deshabilita el botón “Iniciar”. Además, se activa un temporizador que muestra un cronómetro con centésimas de segundo.



- 2) Cada vez que se pulsa el botón “Vuelta”, se obtiene el tiempo acumulado hasta la vuelta en cuestión, y se continua con el cronómetro para la siguiente vuelta:



- 3) Al pulsar el botón “Finalizar”, se para el temporizador, se deshabilitan los botones “Vuelta” y “Finalizar”, y se habilita el botón “Iniciar”.



- 4) Si se sobrepasa el contenido de la caja de texto “tiempos”, se muestra una barra de desplazamiento vertical.

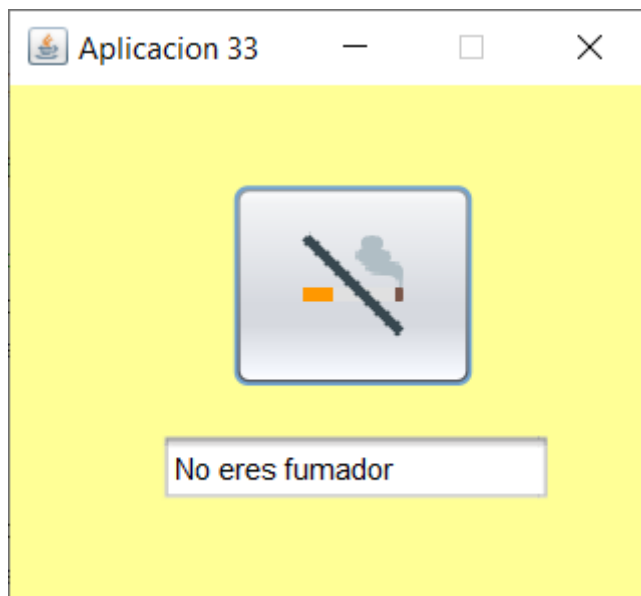
JToggleButton

EVENTOS	
<i>ActionListener</i>	Se lanza el evento cuando se pulsa sobre el control.

PROPIEDADES	
selected	Si está o no activado
buttongroup	Control de agrupamiento
icon	Icono del botón
mnemonic	Tecla de acceso directo
text	Texto actual del control

EJERCICIO

21. Hágase un formulario en el que se tenga un botón de verdadero/falso para indicar si una persona es o no fumador. Si el botón esta activado (verdadero) se mostrará el icono “fumador.png” y si está desactivado, se mostrará el icono “nofumador.png”. Hay una caja de texto de solo lectura cuyo contenido de texto será “Eres fumador” si el botón está activado y “No eres fumador” si no está activado. Inicialmente el botón se muestra desactivado.



- El panel principal tiene un tamaño de 270 x 212
- El botón tiene un tamaño de 90 x 90 con un margen superior de 40, izquierdo y derecha de 90 e inferior de 20
- La caja de texto tiene una anchura de 150 y una altura por defecto (22) y con un margen inferior de 40

CONTROLES DE FORMULARIOS CON MODELO

- Controles de subir y bajar → JSpinner
- Controles de listas → JList
- Controles de tabla → JTable

JSpinner. Uso de clases auxiliares.

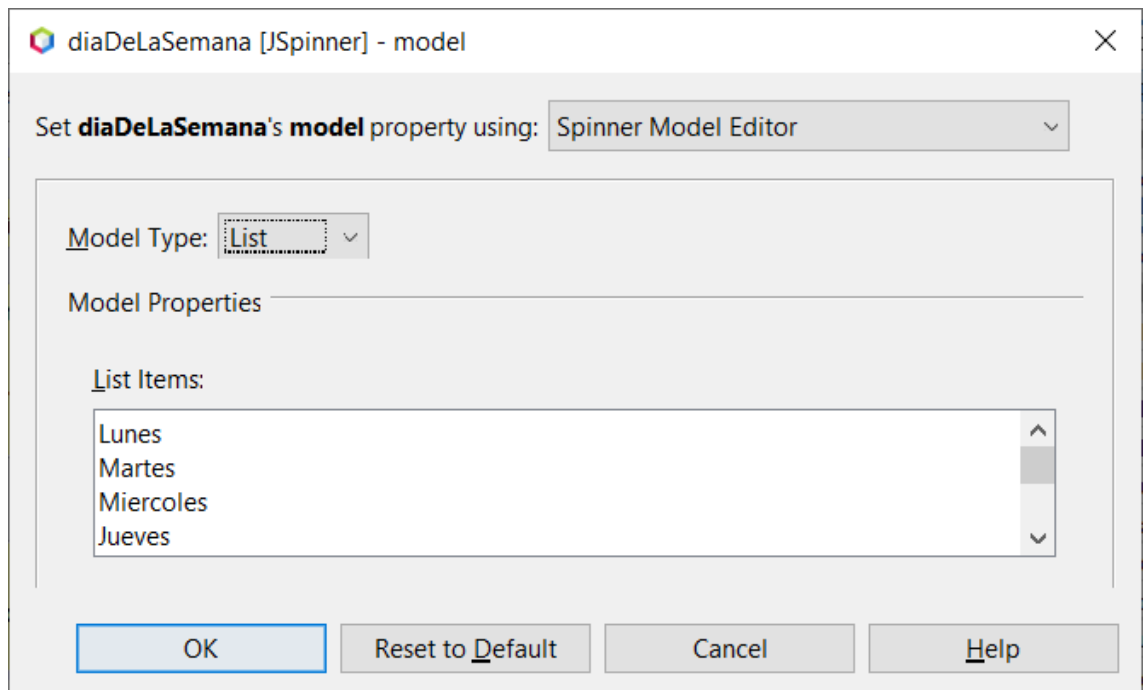
EVENTOS

<i>ChangeListener</i>	Se lanza el evento cuando se modifica el valor.
------------------------------	---

PROPIEDADES

value	Valor actual del control
modelo	Modelo del control (1)

(1) Hay tres tipos de modelos:



precio [JSpinner] - model

Set **precio's model** property using: Spinner Model Editor

Model Type: Number

Model Properties

Number Type: Double

Initial Value: 1

☒ Minimum: 0,5

☒ Maximum: 2,5

Step Size: 0,1

OK Reset to Default Cancel Help

diaDeLaSemana [JSpinner] - model

Set **diaDeLaSemana's model** property using: Spinner Model Editor

Model Type: Date

Model Properties

Initial Value: 23/3/22 22:15 ☒ now

☐ Minimum: 23/3/22 22:15 ☐ now

☐ Maximum: 23/3/22 22:15 ☐ now

Step size: Day of Month

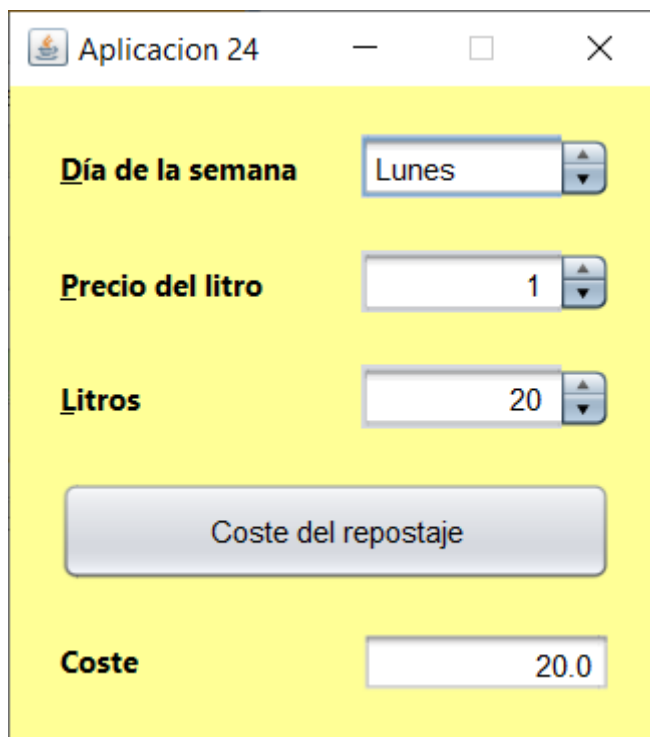
OK Reset to Default Cancel Help

Se puede conseguir que un control numérico de subir/bajar de nombre “precio” sea de solo lectura del siguiente modo:

```
JSpinner.DefaultEditor editorPrecio = (JSpinner.DefaultEditor) precio.getEditor();  
editorPrecio.getTextField().setEditable(false);
```

EJERCICIO

22. Hágase un formulario en el que se introduzcan en controles de subir y bajar con sus correspondientes etiquetas el valor de un repostaje, es decir, el día de la semana, el número de litros (entero entre 1 y 40) y el precio por litro (real entre 0.5 y 2,5). Al pulsar un botón se obtendrá el coste del repostaje en una caja de texto de sólo lectura. La caja de texto de sólo lectura no puede recibir el foco. En cuanto se modifique cualquiera de los valores de los controles subir y bajar se anulará el último coste obtenido de modo que vaciará la caja de texto de sólo lectura y se copiará su valor en el título del formulario.



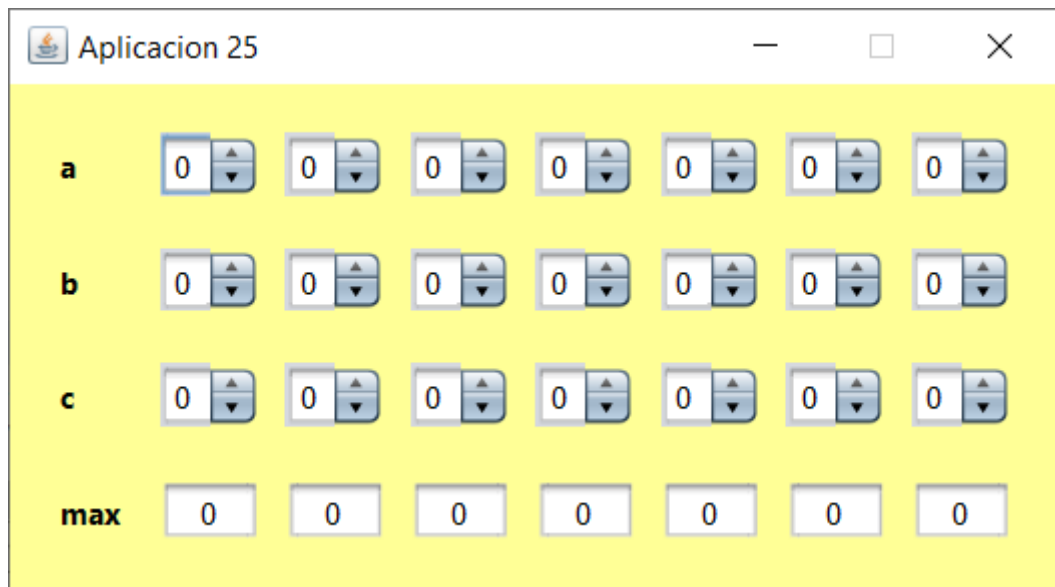
- u. El panel principal tiene un tamaño de 260 x 260 y tiene un relleno de 20
- v. Las etiquetas, los controles de subir/bajar y la caja de texto tienen un tamaño de 100 x 25.
- w. El botón tiene un tamaño de 220 x 40
- x. Todos los controles están separados horizontal y verticalmente 20
- y. La caja de texto es de solo lectura y el texto está alineado a la derecha.
- z. Los controles de subir/bajar son de solo lectura,

NOTA Para vaciar el coste y obtener el título no se utilizan variables de salida

EJERCICIO

23. Se consideran 3 vectores de 7 componentes con valores entre 0 y 9, ambos inclusive. Al modificarse cualquiera de las componentes de alguno de los tres

vectores se obtiene un vector cuyas componentes son el máximo de las componentes de los otros tres vectores.



- aa. El panel principal tiene un tamaño de 420 x 200 y un relleno de 20
- bb. Las etiquetas tienen un tamaño de 30 x 25
- cc. Los controles de subir/bajar y las cajas de texto tienen un tamaño de 40 x 25
- dd. Todos los controles están separados horizontalmente entre sí 10 y verticalmente, 20.
- ee. Las cajas de texto y los controles de subir y bajar son de solo lectura.

JComboBox

EVENTOS	
<i>ItemListener</i>	Se lanza el evento cuando se selecciona un valor del cuadro combinado.

PROPIEDADES	
selectedItem	Valor actual del control seleccionado
editable	De solo lectura o escritura
maximumRowCount	Número máximo de filas que se muestra al desplegar el control
model	Modelo del control (1)
selectedIndex	Índice del elemento actualmente seleccionado

METODOS	
addItem	Añade un nuevo elemento al modelo
insertItemAt	Inserta un elemento en el modelo en una posición dada
removeAllItems	Elimina todos los elementos del modelo
removeItem	Elimina el elemento del modelo

removeItemAt

Elimina el elemento que esta en una determinada posición

(1) Se tiene sólo un modelo:

numeros [JComboBox] - model

Set **numeros's model** property using: Combo Box Model Editor

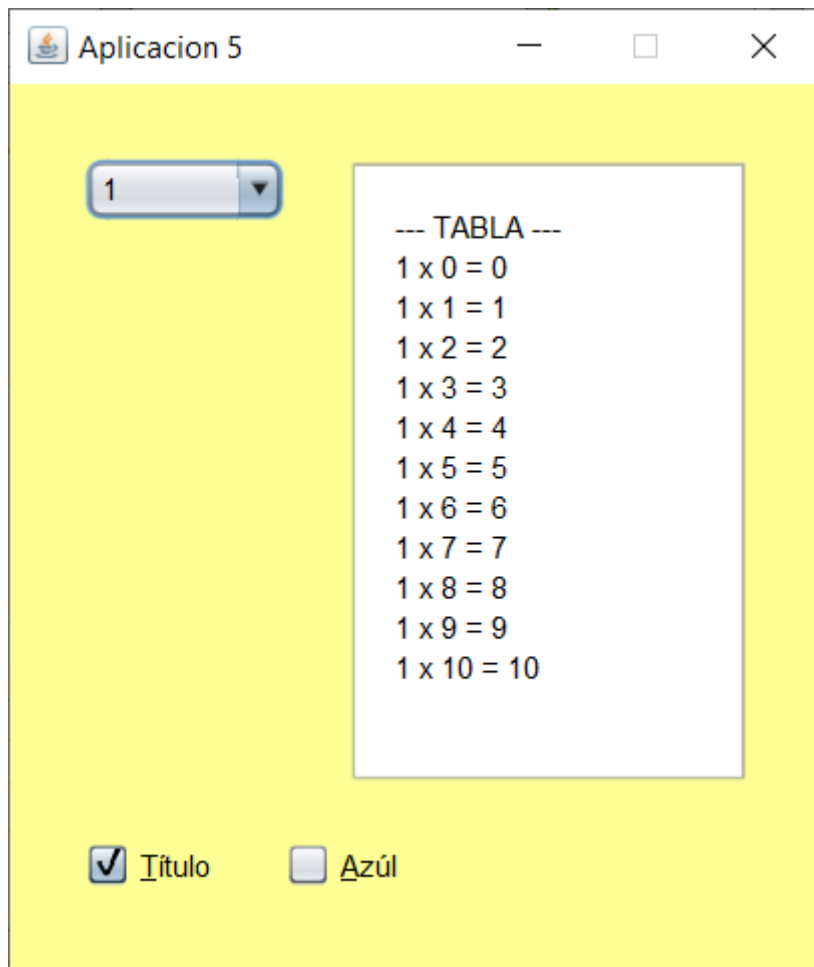
Enter the textual representation of combo box model content. Each row corresponds to one combo box item.

1
2
3
4
5
6
7
8
9
10

OK Reset to Default Cancel Help

EJERCICIO

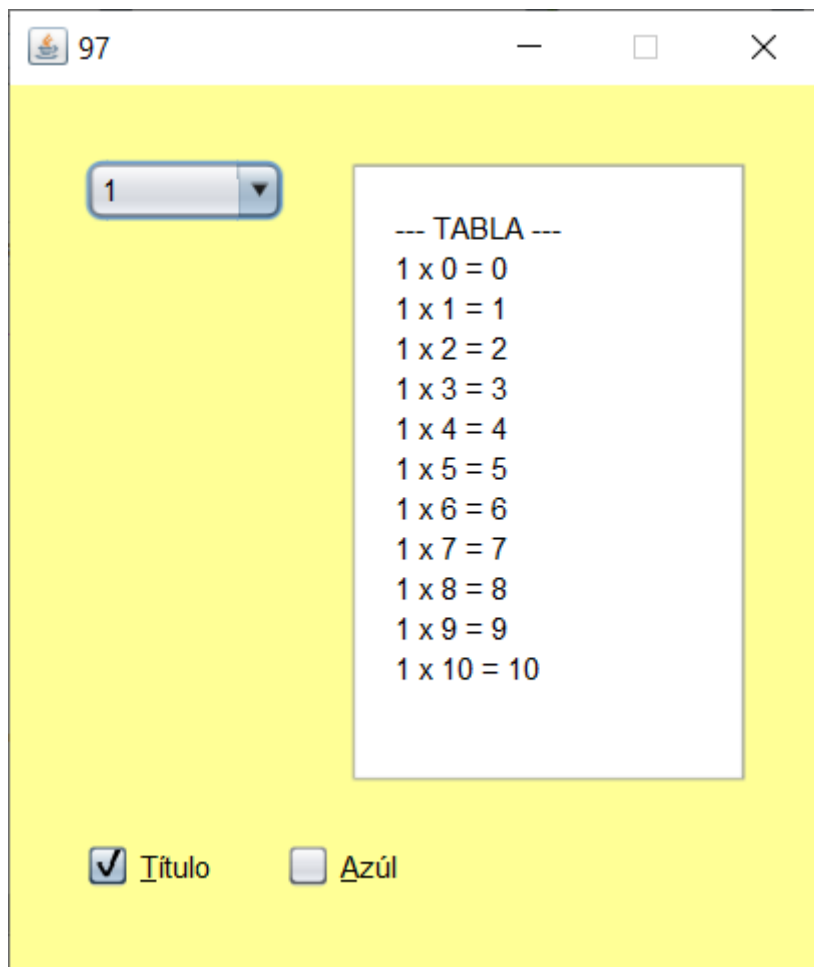
24. Hágase el siguiente formulario:



- a) El panel principal tendrá un tamaño de 325 x 355 con un relleno de 30
 - b) El cuadro combinado tendrá como contenido inicial los enteros 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10. Al menos se mostrarán los 10 primeros elementos al desplegarse. Su tamaño será de 80 x 25. Tendrá un margen derecho de 25
 - c) El panel que contiene la caja de texto multilínea tendrá un tamaño de 160 x 250
 - d) La caja de texto multilínea será de solo lectura y tendrá un margen de 10. Inicialmente se muestra la tabla de multiplicar del 1.
 - e) Las casillas de verificación tendrán un tamaño de 40 x 25 y se separan entre sí y superiormente 20.
- 1) Al seleccionar un elemento del cuadro combinado o al cambiar el estado de "título" o "azul" se obtendrá en la caja de texto multilínea la tabla de multiplicar del número seleccionado, con un título, si "título" estuviera activado y en color azul, si "azul" estuviera activado.

Modifíquese del siguiente modo:

- 2) Al cargarse el formulario se completará el cuadro combinado con valores desde el 11 hasta un número entre 10 y 100, ambos incluidos. El valor máximo se mostrará como título del formulario.

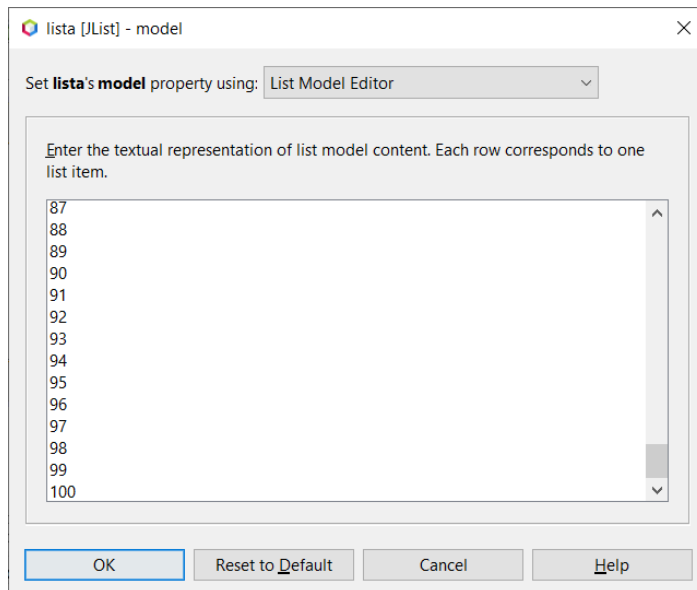


JList

EVENTOS	
<i>ItemListener</i>	Se lanza el evento cuando se selecciona un valor del cuadro combinado.

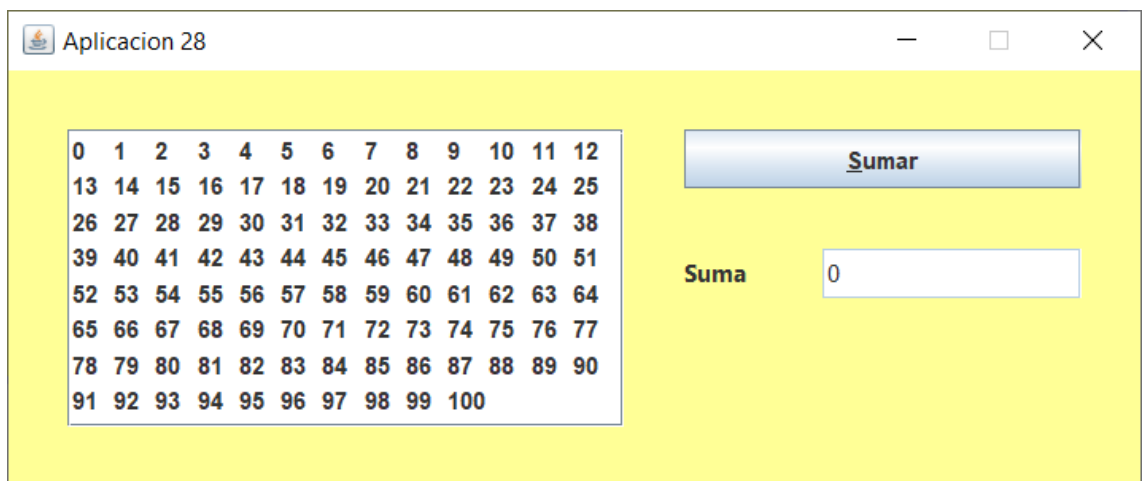
PROPIEDADES	
selectionMode	Modo de selección de elementos
model	Modelo del elemento (1)
layoutOrientation	Como se muestran los valores en la lista
Listas de selección única	
selectedValue	Valor actual del control seleccionado
selectedIndex	Índice del valor actualmente seleccionado
Listas de selección múltiples	
selectedValuesList	Lista de valores actualmente seleccionados
selectedIndices	Índices de los elementos actualmente seleccionados
selectedValues	Valores actuales del control seleccionado

- (1) En la lista se pueden almacenar datos de cualquier dato, pero con el diseñador sólo se permite guardar datos de tipo String



EJERCICIO

25. Hágase un formulario con una lista de selección múltiple que contenga los valores enteros del 0 al 100, ambos incluidos. Al pulsar un botón se obtendrá en una caja de texto de sólo lectura la suma de los elementos seleccionados o 0 si no hubiera ninguno seleccionado. Nada más modificarse la selección de los elementos de la lista, se vaciará la caja de texto de sólo lectura.



- ff. Los controles del formulario se mostrarán con un estilo “Metal”
- gg. El panel principal tendrá un tamaño de 570 x 210 y un relleno de 30
- hh. La lista tendrá un tamaño de 280 x 150. Será de selección múltiple y los elementos se dispondrán en filas
- ii. El botón tendrá un tamaño de 200 x 30
- jj. La etiqueta tendrá un tamaño de 40 x 25

- kk. La caja de texto de sólo lectura con fondo blanco tendrá un tamaño de 130 x 25
- ll. Todos los controles están separados entre sí 30

La clase DefaultListModel

A un control JList se le puede asignar a su atributo “model” un modelo DefaultListModel

Clase DefaultListModel<E>

```
java.lang.Object
    javax.swing.AbstractListModel <E>
        javax.swing.DefaultListModel<E>
```

Constructor y Descripción
DefaultListModel ()

Métodos	
Modificador y Tipo	Método y descripción
void	add (int index, E element) Inserta el elemento especificado en la posición especificada en esta lista.
Métodos	
void	addElement (E element) Agrega el componente especificado al final de esta lista.
int	capacity () Devuelve la capacidad actual de esta lista.
void	clear () Elimina todos los elementos de esta lista.
boolean	contains (Object elem) Comprueba si el objeto especificado es un componente de esta lista.
void	copyInto (Object [] anArray) Copia los componentes de esta lista en la matriz especificada.
E	elementAt (int index) Devuelve el componente en el índice especificado.
Enumeration < E >	elements () Devuelve una enumeración de los componentes de esta lista.
void	ensureCapacity (int minCapacity) Aumenta la capacidad de esta lista, si es necesario, para garantizar que pueda contener al menos el número de componentes especificado por el argumento de capacidad mínima.
E	firstElement () Devuelve el primer componente de esta lista.
E	get (int index) Devuelve el elemento en la posición especificada en esta lista.
E	getElementAt (int index) Devuelve el componente en el índice especificado.
int	getSize () Devuelve el número de componentes de esta lista.
int	indexOf (Object elem) Busca la primera aparición de elem.
int	indexOf (Object elem, int index)

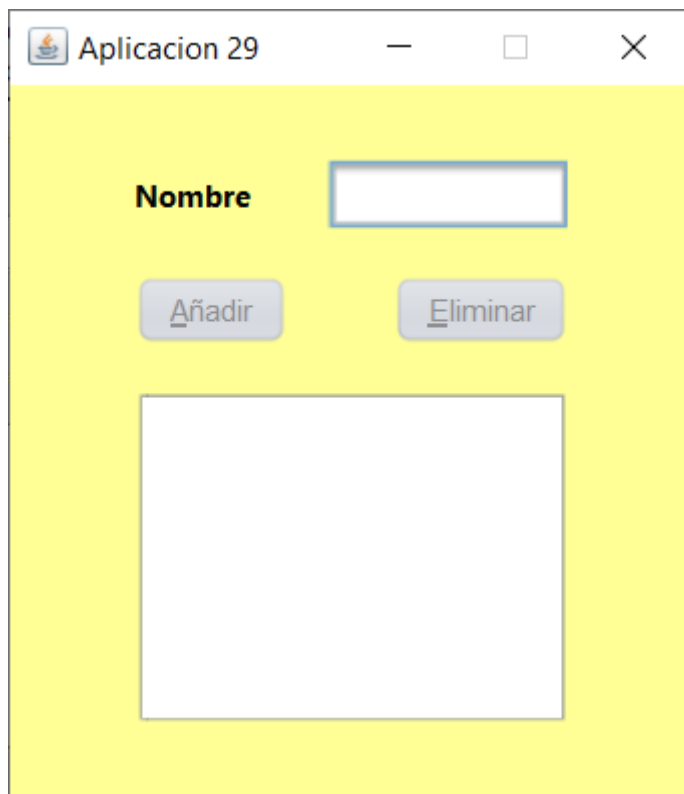
	Busca la primera aparición de <code>elem</code> , comenzando la búsqueda en <code>index</code> .
<code>void</code>	<u>insertElementAt</u> (<code>E element</code> , <code>int index</code>) Inserta el elemento especificado como componente de esta lista en el lugar especificado <code>index</code> .
<code>boolean</code>	<u>isEmpty</u> () Comprueba si esta lista tiene algún componente.
<u>E</u>	<u>lastElement</u> () Devuelve el último componente de la lista.
<code>int</code>	<u>lastIndexOf</u> (<code>Object elem</code>) Devuelve el índice de la última aparición de <code>elem</code> .
<code>int</code>	<u>lastIndexOf</u> (<code>Object elem</code> , <code>int index</code>) Busca hacia atrás <code>elem</code> , a partir del índice especificado, y devuelve un índice.
<u>E</u>	<u>remove</u> (<code>int index</code>) Elimina el elemento en la posición especificada en esta lista.
<code>void</code>	<u>removeAllElements</u> () Elimina todos los componentes de esta lista y establece su tamaño en cero.
<code>boolean</code>	<u>removeElement</u> (<code>Object obj</code>) Elimina la primera ocurrencia (índice más bajo) del argumento de esta lista.
<code>void</code>	<u>removeElementAt</u> (<code>int index</code>) Elimina el componente en el índice especificado.
<code>void</code>	<u>removeRange</u> (<code>int fromIndex</code> , <code>int toIndex</code>) Elimina los componentes en el rango especificado de índices.
<u>E</u>	<u>set</u> (<code>int index</code> , <code>E element</code>) Reemplaza el elemento en la posición especificada en esta lista con el elemento especificado.
<code>void</code>	<u>setElementAt</u> (<code>E element</code> , <code>int index</code>) Establece el componente en el especificado <code>index</code> de esta lista para que sea el elemento especificado.
<code>void</code>	<u>setSize</u> (<code>int newSize</code>) Establece el tamaño de esta lista.
<code>int</code>	<u>size</u> () Devuelve el número de componentes de esta lista.
<code>Object[]</code>	<u>toArray</u> () Devuelve una matriz que contiene todos los elementos de esta lista en el orden correcto.
<u>String</u>	<u>toString</u> () Devuelve una cadena que muestra e identifica las propiedades de este objeto.
<code>void</code>	<u>trimToSize</u> () Recorta la capacidad de esta lista para que tenga el tamaño actual de la lista.
<code>void</code>	<u>addListDataListener</u> (<code>ListDataListener l</code>) Agrega un agente de escucha a la lista que recibe una notificación cada vez que se produce un cambio en el modelo de datos <ul style="list-style-type: none"> • <code>void intervalAdded(ListDataEvent e)</code> • <code>void intervalRemoved(ListDataEvent e)</code> • <code>void contentsChanged(ListDataEvent e)</code>

Eventos	
<code>void</code>	<u>addListDataListener</u> (<code>ListDataListener l</code>)

	Agrega un agente de escucha a la lista que recibe una notificación cada vez que se produce un cambio en el modelo de datos <ul style="list-style-type: none"> • void intervalAdded(ListDataEvent e) • void intervalRemoved(ListDataEvent e) • void contentsChanged(ListDataEvent e)
--	--

EJERCICIO

26. Hágase un formulario que permita añadir y eliminar nombres de personas (palabras que comienzan en mayúsculas y el resto en minúsculas con un máximo de 10 caracteres) de una lista. El nombre a añadir o eliminar se introduce en una caja de texto. Se tendrán dos botones: uno para añadir y otro para eliminar. El botón de añadir estará habilitado si el nombre que se va introduciendo en la caja de texto no existe ya en la lista. El botón de eliminar estará habilitado si el nombre que se va introduciendo en la caja de texto ya existe en la lista.

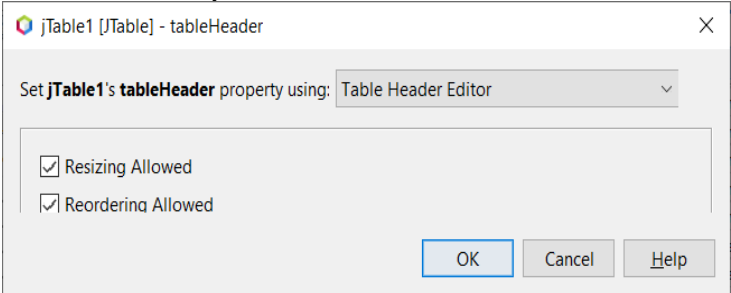


- El panel principal tiene un tamaño de 280 x 300 con un relleno superior e inferior de 30 y de izquierda y derecha de 50
- La etiqueta tiene un tamaño de 50 x 25 con un margen derecho de 20
- La caja de texto tiene un tamaño de 110 x 25. En la cja de texto sólo se pueden introducir nombres de personas con la primera letra en mayúsculas y el resto en minúsculas.
- Los botones tienen un tamaño de 75 x 25 y están separados entre sí 30
- El panel que contiene la lista tiene un tamaño de 180 x 150.
- La lista tiene como modelo el de DefaultListModel

- g. Los controles tienen una separación vertical de 20

JTable

EVENTOS	
<i>ItemListener</i>	Se lanza el evento cuando se selecciona un valor del cuadro combinado.

PROPIEDADES	
autoCreateRowSorter	Poder o no ordenar los datos al pulsar sobre las columnas del control
model	Modelo de la tabla (1)
tableHeader	<p>Establecer si las columnas se pueden redimensionar e intercambiar de posición</p> 
columnModel	Modelo de la columnas
gridColor	Color de la rejilla
showGrid	Mostrar o no la rejilla (líneas horizontales y verticales)
showHorizontalLines	Mostrar las líneas horizontales de la rejilla
showVerticalLines	Mostrar las líneas verticales de la rejilla
rowSelectionAllowed	Seleccionar o no toda la fila al pulsar sobre una celda de esa fila

- (1) Se establecen las columnas y las filas de la tabla

jTable1 [JTable] - model

Set jTable1's model property using:
Table model customizer

Table Model

Table Settings
Default Values

Specify Title and Column Types Here:

Column	Title	Type	Editable
1	Title 1	Object	<input checked="" type="checkbox"/>
2	Title 2	Object	<input checked="" type="checkbox"/>
3	Title 3	Object	<input checked="" type="checkbox"/>
4	Title 4	Object	<input checked="" type="checkbox"/>

Insert

Delete

Move Up

Move Down

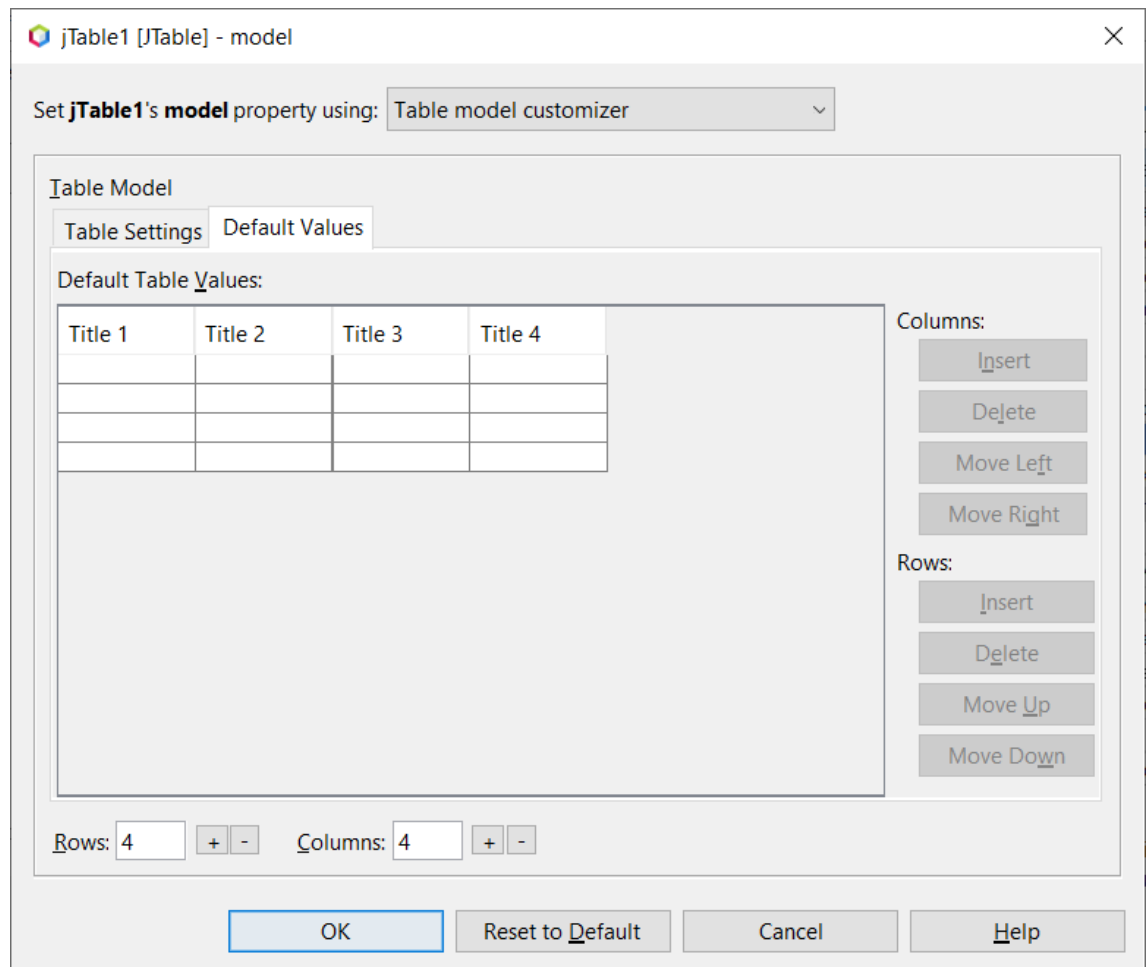
Rows: 4 + -
Columns: 4 + -

OK

Reset to Default

Cancel

Help



A un control JTable se le puede asignar a su atributo “model” un modelo DefaultTableModel

Clase DefaultTableModel

[java.lang.Objeto](#)

[javax.swing.table.AbstractTableModel](#)

[javax.swing.table.DefaultTableModel](#)

Campos	
Modificador y Tipo	Campo y Descripción
protected Vector	columnIdentifiers Los Vectoridentificadores de columna.
protected Vector	dataVector El Vectorde Vectorslos Objectvalores.

Constructor y Descripción	
DefaultTableModel ()	Construye un valor predeterminado DefaultTableModel que es una tabla de cero columnas y cero filas.
DefaultTableModel (int rowCount, int columnCount)	Construye un DefaultTableModelcon rowCounty columnCountde nullvalores de objeto.
DefaultTableModel (Object[][] data, Object [] columnNames)	Construye DefaultTableModele inicializa la tabla pasando datay columnNames al setDataVector método.

DefaultTableModel (Object [] columnNames, int rowCount)
Construye un DefaultTableModel con tantas columnas como elementos hay en columnNames y rowCount de null valores de objeto.
DefaultTableModel (Vector columnNames, int rowCount)
Construye un DefaultTableModel con tantas columnas como elementos hay en columnNames y rowCount de null valores de objeto.
DefaultTableModel (Vector data, Vector columnNames)
Construye DefaultTableModel e inicializa la tabla pasando data y columnNames al setDataVector método.

Métodos	
Modificador y Tipo	Método y descripción
void	addColumn (Object columnName) Añade una columna al modelo.
void	addColumn (Object columnName, Object [] columnData) Añade una columna al modelo.
void	addColumn (Object columnName, Vector columnData) Añade una columna al modelo.
void	addRow (Object [] rowData) Añade una fila al final del modelo.
void	addRow (Vector rowData) Añade una fila al final del modelo.
protected static Vector	convertToVector (Object [] anArray) Devuelve un vector que contiene los mismos objetos que la matriz.
protected static Vector	convertToVector (Object [][] anArray) Devuelve un vector de vectores que contiene los mismos objetos que la matriz.
int	getColumnCount () Devuelve el número de columnas de esta tabla de datos.
String	columnName (int column) Devuelve el nombre de la columna.
Vector	getDataVector () Devuelve el Vector de Vectors que contiene los valores de datos de la tabla.
int	getRowCount () Devuelve el número de filas de esta tabla de datos.
Object	getValueAt (int row, int column) Devuelve un valor de atributo para la celda en row y column.
void	insertRow (int row, Object [] rowData) Inserta una fila row en el modelo.
void	insertRow (int row, Vector rowData) Inserta una fila row en el modelo.
boolean	isCellEditable (int row, int column) Devuelve verdadero independientemente de los valores de los parámetros.
void	moveRow (int start, int end, int to) Mueve una o más filas del rango inclusivo start a end a la posición en el modelo.
void	newDataAvailable (TableModelEvent event) equivalente a fireTableChanged.
void	newRowsAdded (TableModelEvent e) Garantiza que las nuevas filas tengan el número correcto de columnas.
void	removeRow (int row) Elimina la fila en row del modelo.
void	rowsRemoved (TableModelEvent event) equivalente a fireTableChanged.
void	setColumnCount (int columnCount) Establece el número de columnas en el modelo.
void	setColumnIdentifiers (Object [] newIdentifiers) Reemplaza los identificadores de columna en el modelo.
void	setColumnIdentifiers (Vector columnIdentifiers)

	Reemplaza los identificadores de columna en el modelo.
void	<u>setDataVector</u> (<u>Object</u> [][] dataVector, <u>Object</u> [] columnIdentifiers) Reemplaza el valor de la dataVector variable de instancia con los valores de la matriz dataVector.
void	<u>setDataVector</u> (<u>Vector</u> dataVector, <u>Vector</u> columnIdentifiers) Reemplaza la dataVector variable de instancia actual con la nueva Vector de filas, dataVector.
void	<u>setNumRows</u> (int rowCount) Obsoleto a partir de la plataforma Java 2 v1.3.
void	<u>setRowCount</u> (int rowCount) Establece el número de filas en el modelo.
void	<u>setValueAt</u> (<u>Object</u> aValue, int row, int column) Establece el valor del objeto para la celda en column y row.

Eventos	
void	<u>addTableModelListener</u> (<u>TableModelListener</u> l) Agrega un agente de escucha a la lista que recibe una notificación cada vez que se produce un cambio en el modelo de datos

Clase DefaultTableModel

java.lang.Objeto

javax.swing.table.DefaultTableModel

Modificador y Tipo	Campo
protected <u>ChangeEvent</u>	<u>changeEvent</u> Cambiar evento (solo se necesita uno)
protected int	<u>columnMargin</u> Ancho margen entre cada columna
protected boolean	<u>columnSelectionAllowed</u> Selección de columna permitida en este modelo de columna
protected <u>EventListenerList</u>	<u>listenerList</u> Lista de TableColumnModelListener
protected <u>ListSelectionModel</u>	<u>selectionModel</u> Modelo para realizar un seguimiento de las selecciones de columnas
protected <u>Vector</u> < <u>TableColumn</u> >	<u>tableColumns</u> Matriz de objetos TableColumn en este modelo
protected int	<u>totalColumnWidth</u> Un caché local del ancho combinado de todas las columnas

Constructor	Descripción
<u>DefaultTableModel</u> ()	Crea un modelo de columna de tabla predeterminado.

Modificador y Tipo	Método/Descripción
void	<u>addColumn</u> (<u>TableColumn</u> aColumn) Se agrega aColumnal final de la tableColumnsmatriz
void	<u>addColumnModelListener</u> (<u>TableColumnModelListener</u> x) Agrega un detector para eventos de modelo de columna de tabla.
protected <u>ListSelectionModel</u>	<u>createSelectionModel</u> () Crea un nuevo modelo de selección de lista predeterminado.
protected void	<u>fireColumnAdded</u> (<u>TableColumnModelEvent</u> e)

	Notifica a todos los oyentes que han registrado interés para recibir notificaciones sobre este tipo de evento.
protected void	<u>fireColumnMarginChanged</u> () Notifica a todos los oyentes que han registrado interés para recibir notificaciones sobre este tipo de evento.
protected void	<u>fireColumnMoved</u> (<u>TableColumnModelEvent</u> e) Notifica a todos los oyentes que han registrado interés para recibir notificaciones sobre este tipo de evento.
protected void	<u>fireColumnRemoved</u> (<u>TableColumnModelEvent</u> e) Notifica a todos los oyentes que han registrado interés para recibir notificaciones sobre este tipo de evento.
protected void	<u>fireColumnSelectionChanged</u> (<u>ListSelectionEvent</u> e) Notifica a todos los oyentes que han registrado interés para recibir notificaciones sobre este tipo de evento.
<u>TableColumn</u>	<u>getColumn</u> (int columnIndex) Devuelve el <u>TableColumn</u> objeto de la columna en columnIndex.
int	<u>getColumnCount</u> () Devuelve el número de columnas de la <u>tableColumns</u> matriz.
int	<u>getColumnIndex</u> (<u>Object</u> identifier) Devuelve el índice de la primera columna de la <u>tableColumns</u> matriz cuyo identificador es igual a identifier, cuando se compara con equals.
int	<u>getColumnIndexAtX</u> (int x) Devuelve el índice de la columna que se encuentra en la posición x o - 1 si ninguna columna cubre este punto.
int	<u>getColumnMargin</u> () Devuelve el margen de ancho para <u>TableColumn</u> .
<u>TableColumnModelListener</u> []	<u>getColumnModelListeners</u> () Devuelve una matriz de todos los detectores del modelo de columna registrados en este modelo.
<u>Enumeration</u> < <u>TableColumn</u> >	<u>getColumns</u> () Devuelve una <u>Enumeration</u> de todas las columnas del modelo.
boolean	<u>getColumnSelectionAllowed</u> () Devuelve verdadero si se permite la selección de columnas; de lo contrario, devuelve falso.
<T extends <u>EventListener</u> > T []	<u>getListeners</u> (<u>Class</u> <T> listenerType) Devuelve una matriz de todos los objetos registrados actualmente como <u>FooListeners</u> en este modelo.
int	<u>getSelectedColumnCount</u> () Devuelve el número de columnas seleccionadas.
int []	<u>getSelectedColumns</u> () Devuelve una matriz de columnas seleccionadas.
<u>ListSelectionModel</u>	<u>getSelectionModel</u> () Devuelve el <u>ListSelectionModel</u> que se utiliza para mantener el estado de selección de columna.
int	<u>getTotalColumnWidth</u> () Devuelve el ancho total combinado de todas las columnas.
void	<u>moveColumn</u> (int columnIndex, int newIndex) Mueve la columna y el encabezado de columnIndex a newIndex.
void	<u>propertyChange</u> (<u>PropertyChangeEvent</u> evt) Método de cambio de escucha de cambio de propiedad.
protected void	<u>recalcWidthCache</u> () Vuelve a calcular el ancho total combinado de todas las columnas.
void	<u>removeColumn</u> (<u>TableColumn</u> column) Elimina el column de la <u>tableColumns</u> matriz.

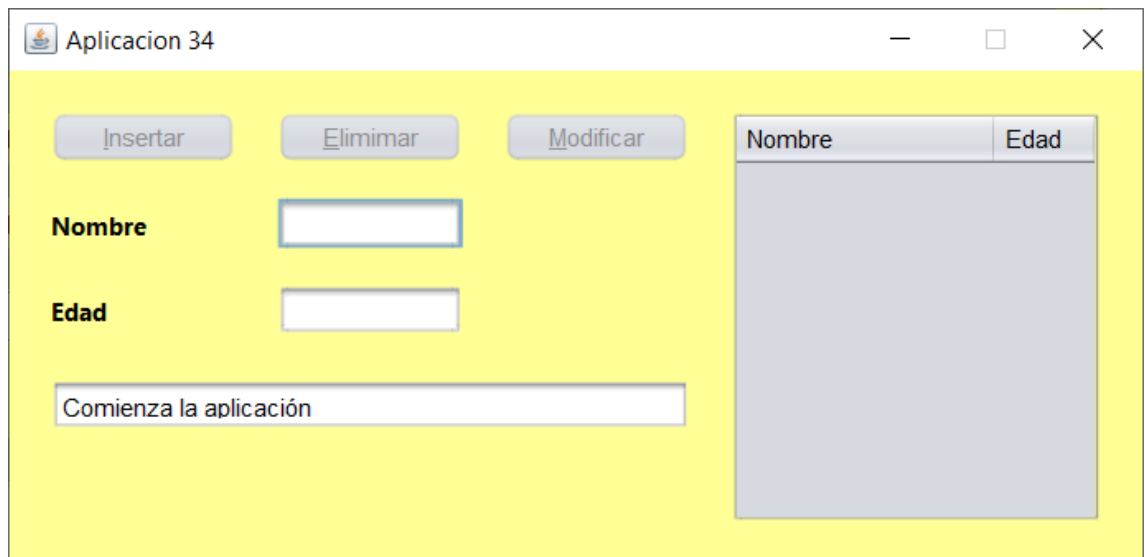
void	<u>removeColumnModelListener</u> (<u>TableColumnModelListener</u> x) Elimina un detector para eventos de modelo de columna de tabla.
void	<u>setColumnMargin</u> (int newMargin) Establece el margen de la columna en newMargin.
void	<u>setColumnSelectionAllowed</u> (boolean flag) Establece si se permite la selección de columnas.
void	<u>setSelectionModel</u> (<u>ListSelectionModel</u> newModel) Establece el modelo de selección para esto TableColumnModel y se newModel registra para las notificaciones del oyente del nuevo modelo de selección.
void	<u>valueChanged</u> (<u>ListSelectionEvent</u> e) ListSelectionListener que reenvía ListSelectionEvents cuando hay un cambio de selección de columna.

Establecer el ancho de las columnas

```
tabla.getColumnModel().getColumn(0).setPreferredWidth(130);
tabla.getColumnModel().getColumn(1).setPreferredWidth(50);
```

EJERCICIO

- 27.** Hágase un formulario que permita añadir, eliminar y modificar personas (nombres y edades) en una tabla. Los nombres de las personas se introducirán en una caja de texto y serán palabras que comienzan en mayúsculas y el resto en minúsculas con un máximo de 10 caracteres). La edad se introducirá en otra caja de texto y será o la cadena vacía o un valor entre 0 y 120, ambos inclusive. La inserción, eliminación y modificación de las personas se realizarán al pulsar los botones de inserción, eliminación y modificación, respectivamente. El botón de inserción estará habilitado si el nombre de la persona es distinto de la cadena vacía y no existe en la tabla. Los botones de eliminar y modificar estarán habilitados si el nombre de la persona existe en la tabla. Al eliminar una persona de la tabla se mostrará en la caja de texto que contiene la edad, la edad de la persona eliminada. Se tendrá una caja de estado de solo lectura que mostrará la última acción realizada: “Comienza la aplicación”; “Nombre modificado”, “Edad modificada”; “Persona insertada”, “Persona eliminada” y “Persona modificada”



- a) El panel principal tendrá un tamaño de 550 x 240
- b) Los botones, las etiquetas y las cajas de texto de introducción de datos tendrán un tamaño de 90 x 25
- c) La caja de texto de sólo lectura tendrá un tamaño de 290 x 25
- d) El panel que contiene la tabla tendrá un tamaño de 180 x 200
- e) En la tabla las columnas no se pueden intercambiar ni redimensionar. las columnas La primera columna tendrá una anchura de 130 y la segunda de 50. No se podrán seleccionar las filas.
- f) Todos los controles están separados entre sí y del panel principal 20
- g) El texto de la caja donde se introduce la edad estará alienado a la derecha.
- h) Sólo pueden recibir el foco con el tabulador las cajas de texto de “nombre” y “edad”.

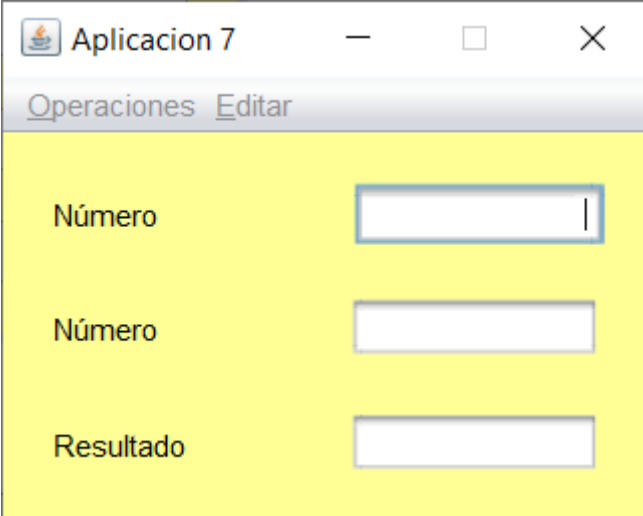
MENUS, CUADROS DE DIALOGO, MULTIPLES FORMULARIOS, FORMULARIOS MODALES Y MENU DE FORMULARIOS

- Barra de menús → JBarMenu
- Elemento de un menú → JMenuItem
- Submenús → JMenu
- Menus de formularios → PLANTILLA_MENU
- Cuadro de dialogo de confirmación → JOptionPane.showConfirmDialog
- Cuadro de dialogo de entrada → JOptionPane.showInputDialog
- Cuadro de dialogo de mensajes → JOptionPane.showMessageDialog
- Cuadro de dilogos de color → JColorChooser.showDialog
- Formularios modales → JDialog

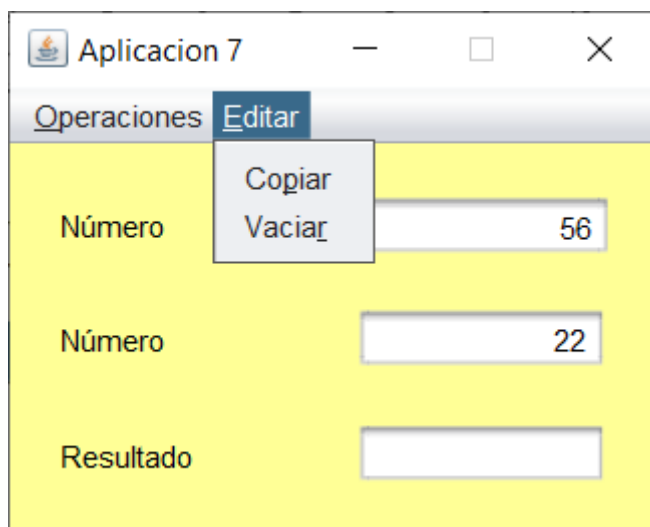
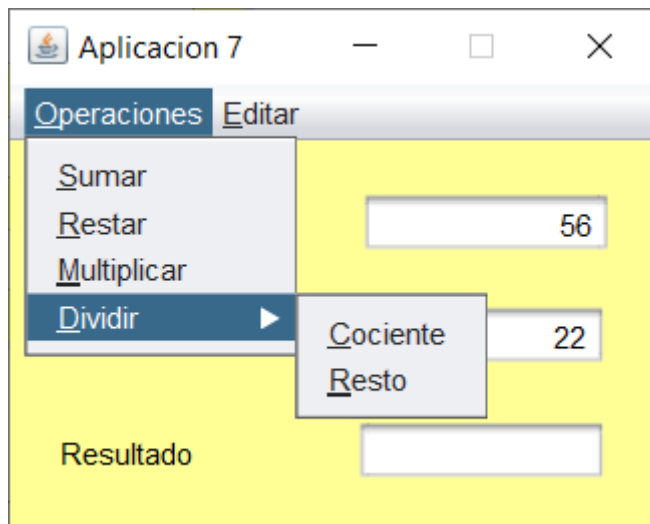
JMenuBar, JMenuItem y JMenu

EJERCICIO

28. Hágase un formulario que tenga una barra de menú con los menús “Operaciones” y “Editar”. “Operaciones” tendrá cuatro opciones “Sumar”, “Restar”, “Multiplicar” y “Dividir”. Ésta última será un submenú con las opciones “Cociente” y “Resto”. El menú “Editar” tendrá las opciones de “Copiar” y “Vaciar”.



The screenshot shows a Java Swing window titled "Aplicacion 7". The window has a standard title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with two items: "Operaciones" and "Editar". The main content area has a yellow background. It contains three labels with corresponding input fields: "Número" (with a small blue border), "Número", and "Resultado".



- El panel principal tendrá un tamaño de 300 x 300
- Todas las etiquetas, cajas de texto de introducción de datos y caja de texto de sólo lectura tendrán un tamaño de 100 x 25.
- Todos los controles están separados entre sí 20
- Las cajas de texto están alineadas a la derecha.
- En las dos primeras cajas de texto sólo se pueden introducir número naturales entre 0 y 99999, ambos inclusive y la cadena vacía.
- El menú operaciones estará deshabilitado si alguna de las cajas de texto de introducción de datos está vacía.
- El submenú “dividir” estará deshabilitado si el contenido de la segunda caja de texto es la cadena vacía o el 0”.
- El menú editar está deshabilitado si las dos cajas de texto de introducción de datos están vacías.
- La opción copiar está deshabilitada si la primera caja de texto de introducción de datos está vacía.
- Al pulsar el botón de “sumar”, “restar”, “multiplicar”, “cociente” y “resto” se obtendrá dicha operación para los dos datos introducidos, se mostrará en la

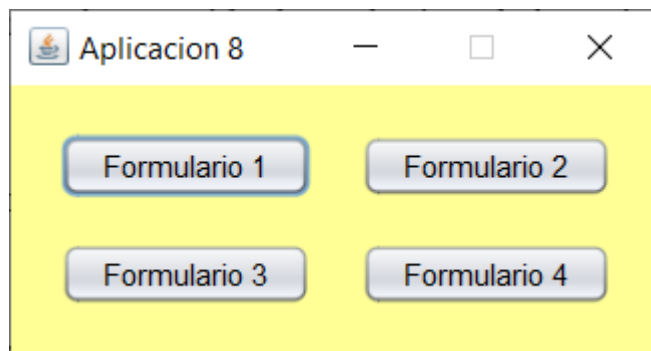
caja de texto de sólo lectura y en su etiqueta se pondrá “Suma”, “Resta”, “Producto”, “Cociente” o “Resto”, respectivamente.

- 2) Al pulsar el botón de “copiar” se copiará el contenido de la primera caja de texto en la caja de sólo lectura y se pondrá en su etiqueta “Copia”
- 3) Al pulsar el botón de “vaciar” se vacían todas las cajas de texto y en la etiqueta asociada a la caja de sólo lectura se pondrá “Resultado”.
- 4) Al modificarse el contenido de cualquiera de las cajas de texto de introducción de datos, se habilitarán los botones del menú, se vaciará la caja de texto de sólo lectura y en su etiqueta se pondrá “Resultado”

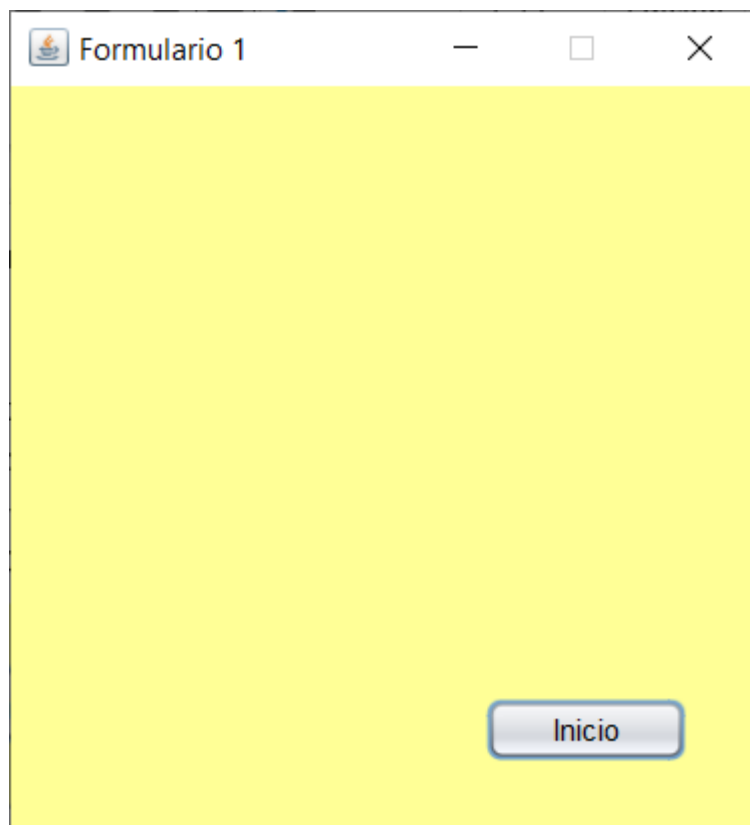
Múltiples formularios

EJERCICIO

29. Hágase un formulario que tenga cuatro botones. Al pulsar cada botón se cerrará el formulario actual y se mostrará un nuevo formulario.



Al pulsar el botón “Formulario 1” se muestra el formulario 1



Al pulsar el botón “inicio” en cada uno de los cuatro formularios, se cierra el formulario actual y se muestra el formulario de inicio.

- a) El panel principal del formulario inicial tiene un tamaño de 260 x 110 con un relleno de 20
- b) Los botones tienen un tamaño de 100 x 25 y están separados horizontal y verticalmente 20
- c) Los paneles principales de los cuatro formularios tienen un tamaño de 300 x 300. Contienen un botón de 80 x 25 con el texto de “Inicio” que tiene un margen derecho e inferior de 30. El título de cada uno de tales formularios es “Formulario 1”, “Formulario 2”, “Formulario 3” y “Formulario 4”

Modificar el procedimiento principal de un formulario auxiliar

```
public static void inicio() {  
  
    ...  
  
}
```

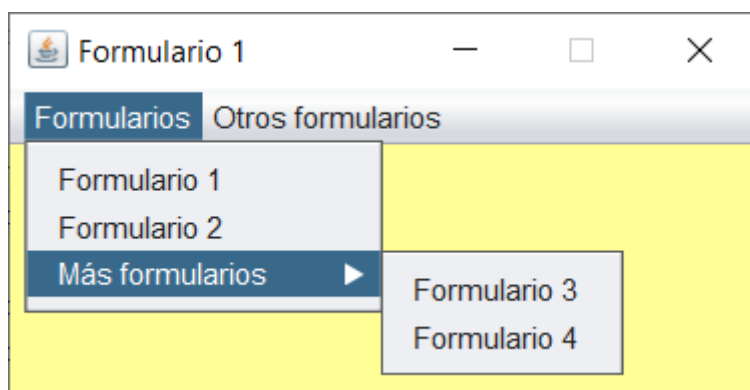
Cerrar el formulario actual y mostrar un nuevo formulario

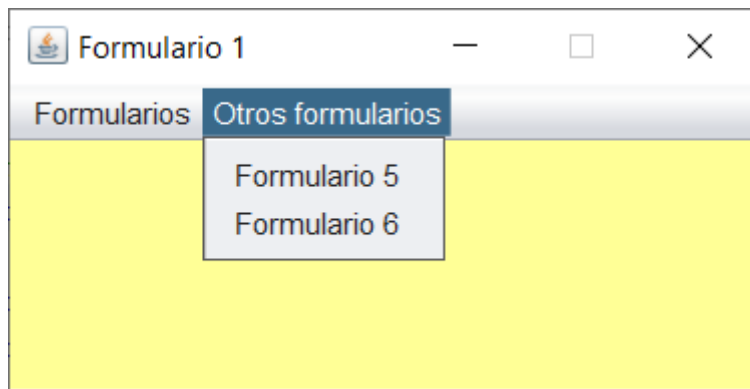
```
v.dispose();  
Vista1.inicio();
```

Menú de formularios. Uso de la PLANTILLA_MENU

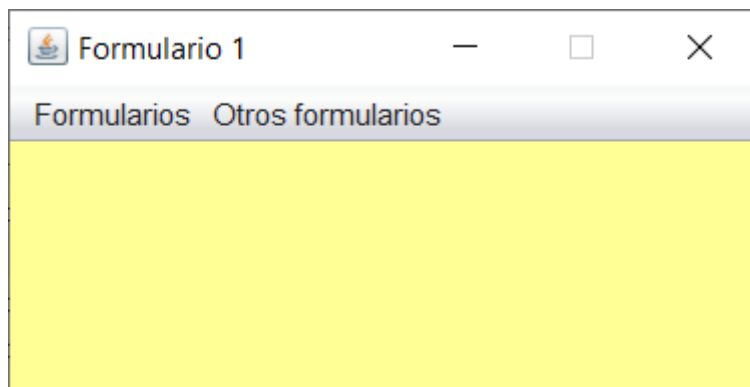
EJERCICIO (ver la plantilla menu)

30. Hágase un formulario que permita mostrar 6 formularios distintos a partir de botones almacenados en un menú como se muestra en la imagen

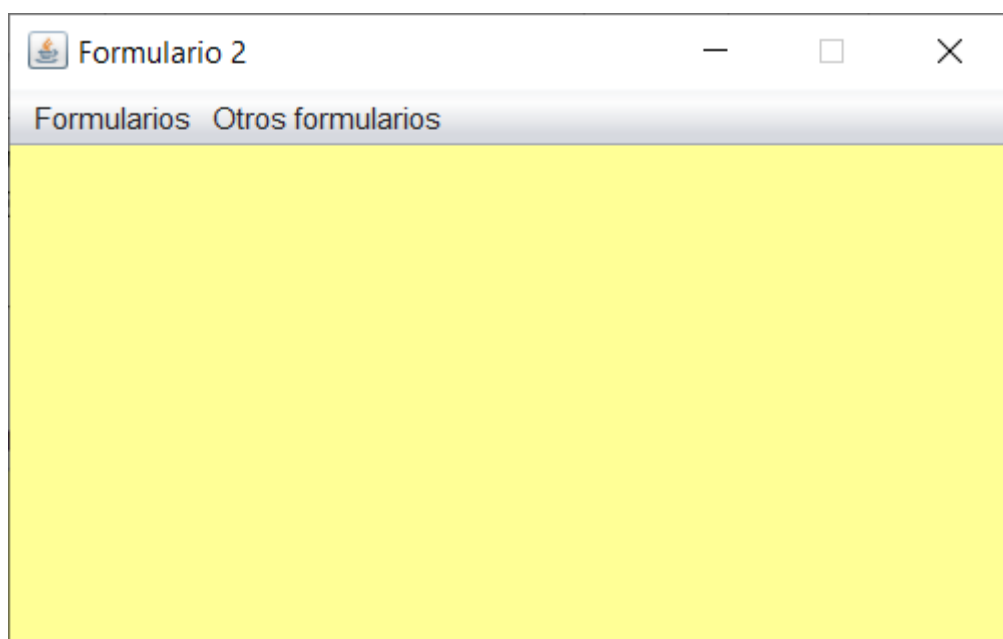




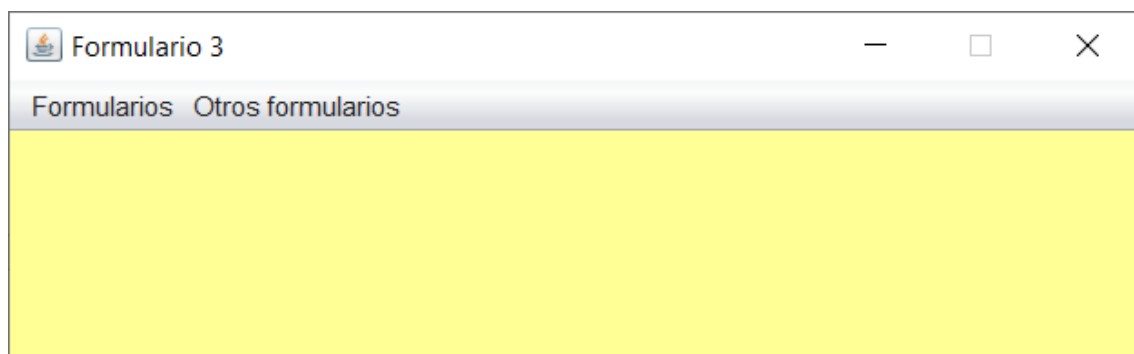
- a) Los seis formularios tendrán como título “Formulario 1”, “Formulario 2”, “Formulario 3”, “Formulario 4”, “Formulario 5” y “Formulario 6” y por supuesto se mostrarán centrados en la pantalla
- b) El tamaño del panel de cada formulario (del 1 al 6) serían 300 x 100, 400 x 200, 500 x 100, 300 x 400, 400 x 300 y 500 x 200.



300 x 100

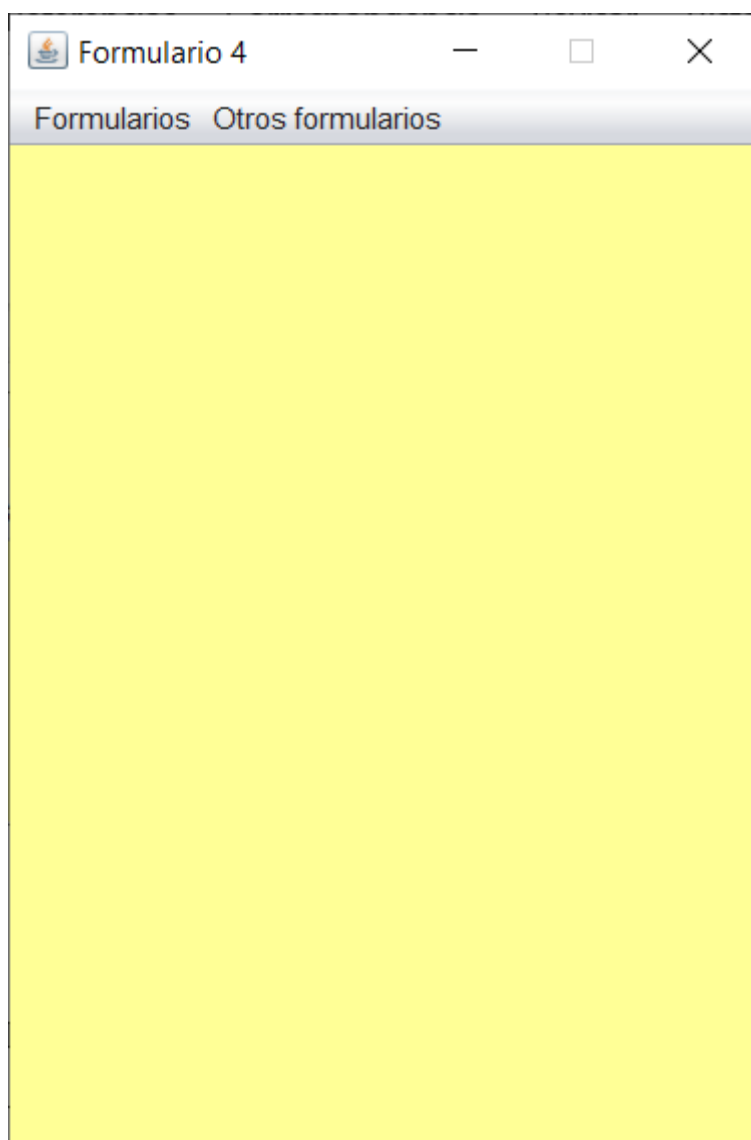


400 x 200



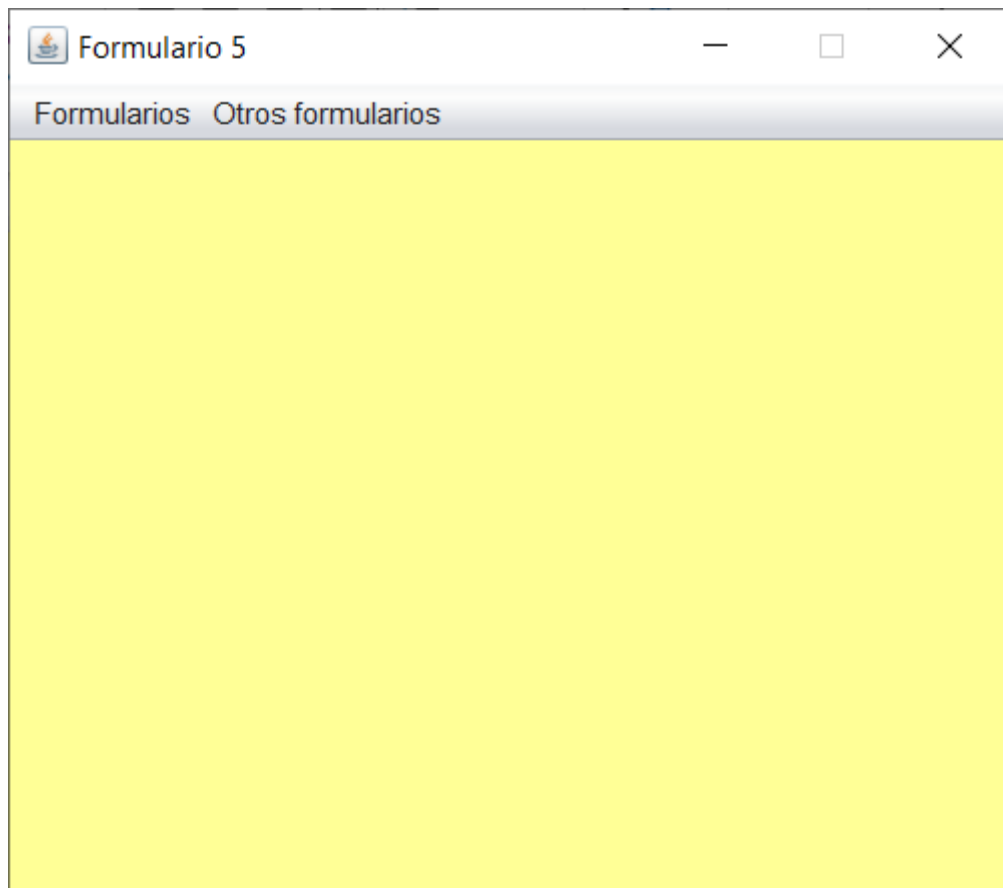
A screenshot of a web browser window. The title bar shows a small icon followed by the text 'Formulario 3'. To the right of the title are standard window controls: a minus sign, a square, and an 'X'. Below the title bar is a navigation bar with two links: 'Formularios' and 'Otros formularios'. The main content area of the browser is a solid yellow rectangle.

500 x 100

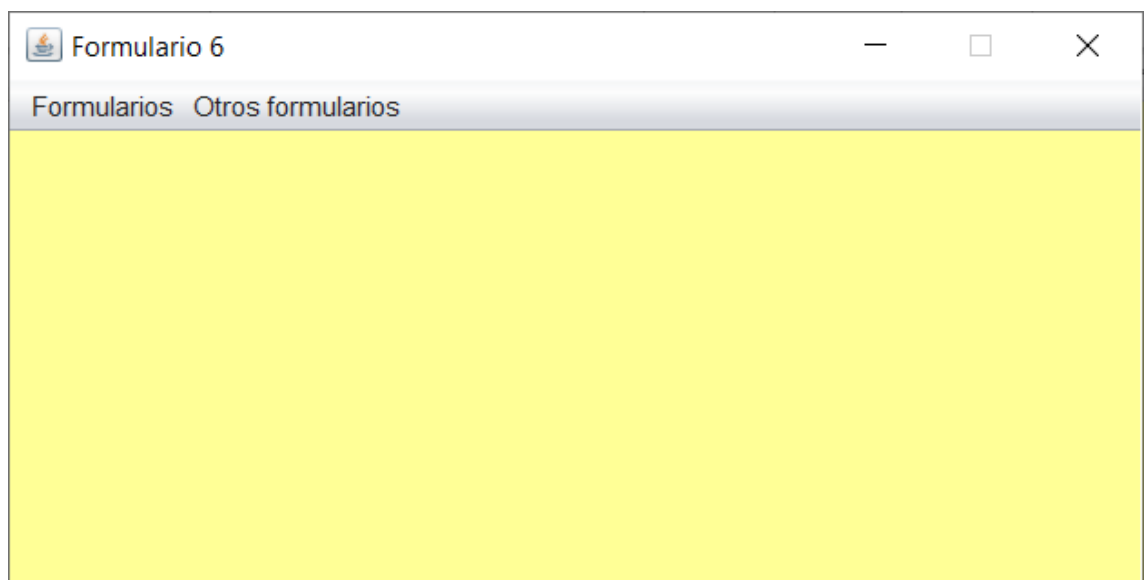


A screenshot of a web browser window. The title bar shows a small icon followed by the text 'Formulario 4'. To the right of the title are standard window controls: a minus sign, a square, and an 'X'. Below the title bar is a navigation bar with two links: 'Formularios' and 'Otros formularios'. The main content area of the browser is a solid yellow rectangle.

300 x 400



400 x 300

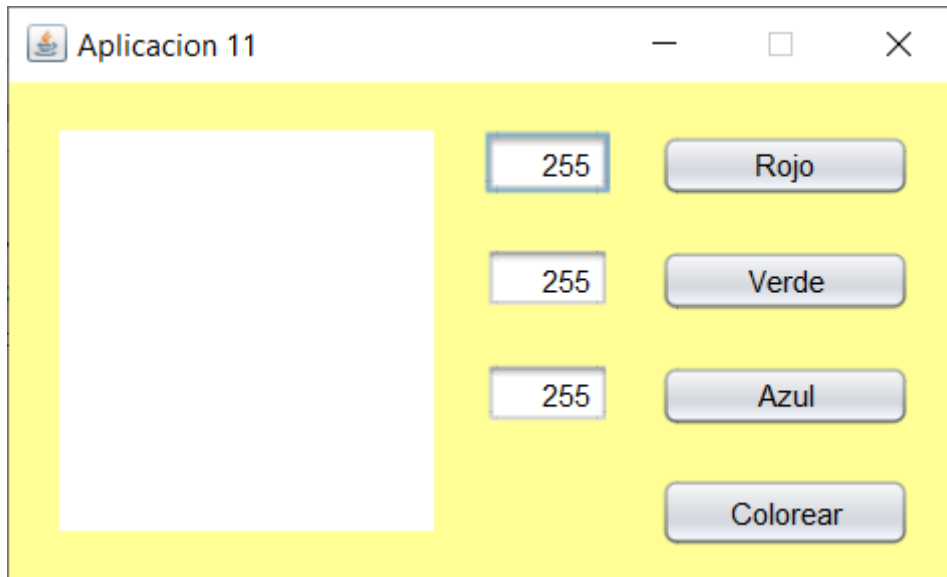


500 x 200

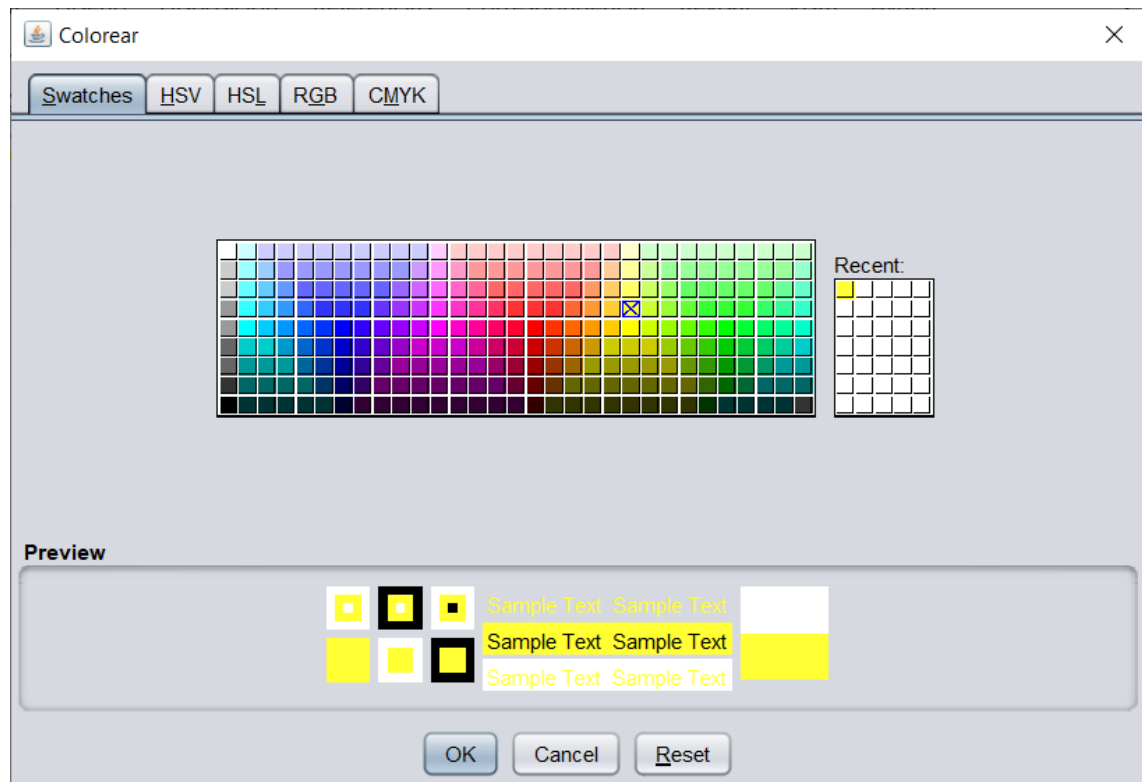
Cuadros de diálogos

EJERCICIO (ver cómo se muestran los cuadros de diálogos y se comprueba que se hayan cancelado)

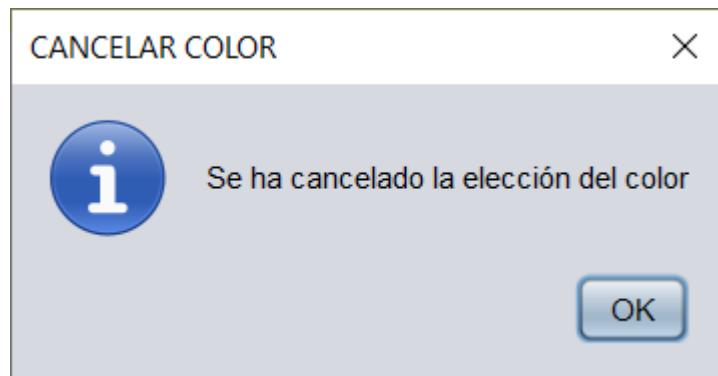
31. Hágase un formulario que permita modificar el fondo de color de un panel estableciendo el color o modificando cada uno de los matices de los colores rojo, verde y azul.



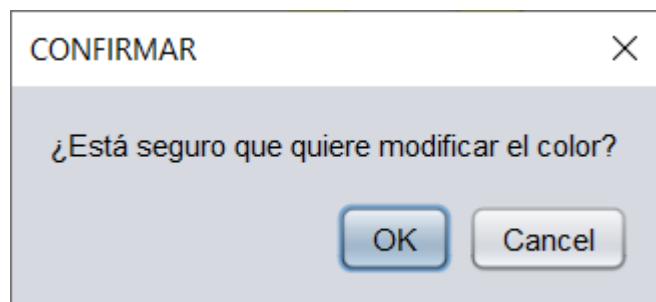
- mm. El panel principal tendrá un tamaño de 300 x 300 con un margen de 20
 - nn. El panel (cuyo fondo se colorea) tiene un tamaño de 150 x 160
 - oo. Las cajas de texto tienen un tamaño de 50 x 25, son de solo lectura, con fondo blanco y el texto alineado a la derecha.
 - pp. Los botones tienen un tamaño de 100 x 25
 - qq. Todos los controles están separados entre sí 20
- 1) Al pulsar el botón de colorear se mostrará un cuadro de dialogo de color para seleccionar el color.



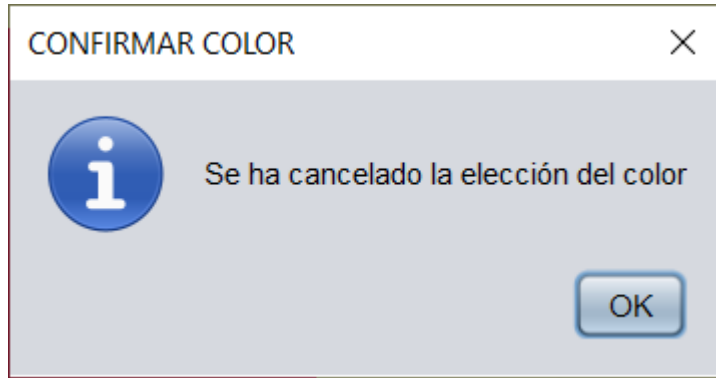
Se podrá cancelar la asignación del color mostrando un cuadro de dialogo de mensaje indicando tal hecho.



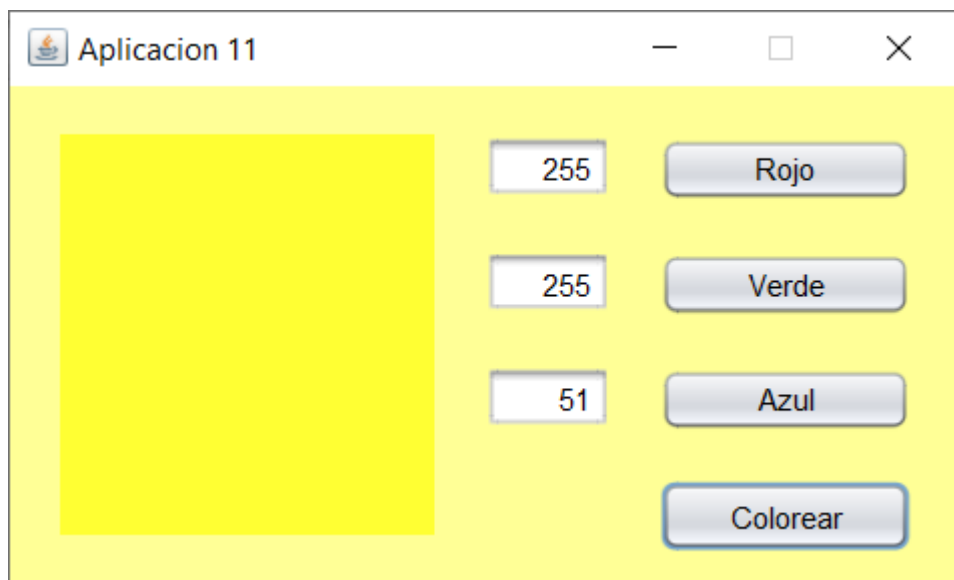
Si se selecciona un color, se muestra un cuadro dialogo de confirmación.



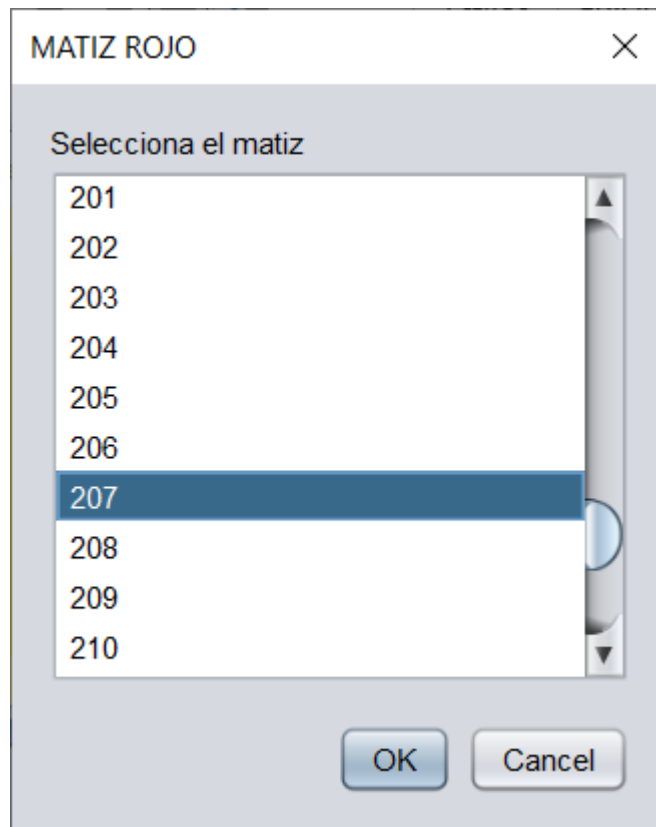
De cancelarse la asignación del color, se muestra un cuadro de dialogo inidicándolo



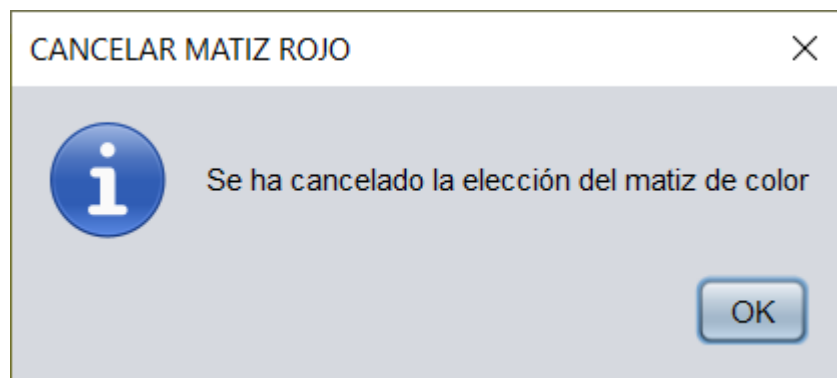
; si no, se establece el fondo del color del panel y se actualizan las cajas de texto de los matices de rojo, verde y azul.



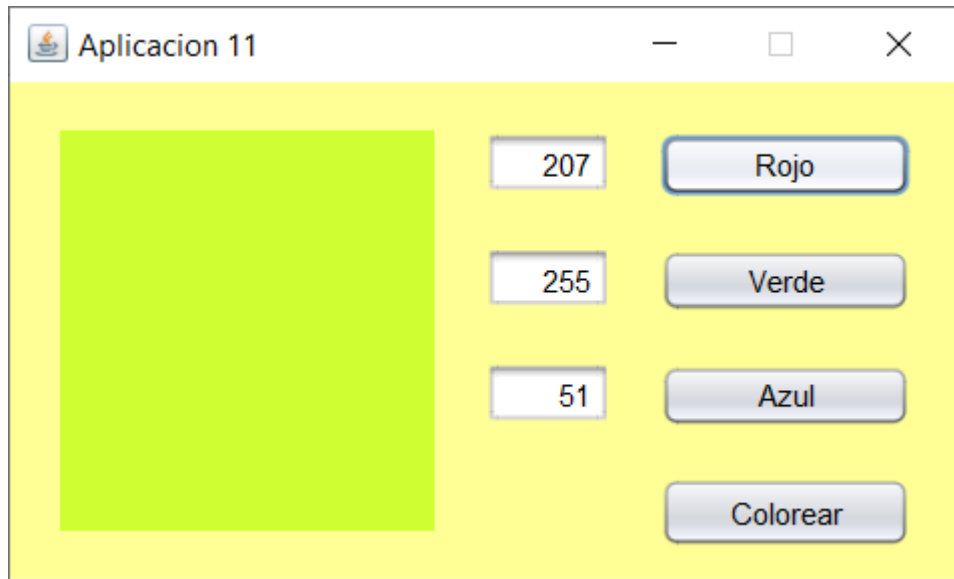
- 2) Al pulsar cualquiera de los botones “rojo”, “verde” y “azul” se mostrará un cuadro de dialogo de entrada con los valores de 0 a 255 y seleccionado el valor de 0



De cancelarse la elección del matiz, se muestra un cuadro de dialogo de mensaje



De no cancelarse la elección del matiz, se actualiza la correspondiente caja de texto y el color del panel



Mostrar un cuadro de dialogo de color y comprobar si se ha cancelado la elección

```
Color c = JColorChooser.showDialog(v, "Colorear", Color.white);
if(c==null)
{
}
```

Mostrar un cuadro de dialogo de mensaje

```
JOptionPane.showMessageDialog(v, "Se ha cancelado la elección del
color", "CANCELAR COLOR", JOptionPane.INFORMATION_MESSAGE, null);
```

Mostrar un cuadro de dialogo de confirmación y comprobar si se ha cancelado

```
int n = JOptionPane.showConfirmDialog(v, "¿Está seguro que quiere
modificar el color?","CONFIRMAR", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.PLAIN_MESSAGE,null);
if(JOptionPane.CANCEL_OPTION==n)
{
}
```

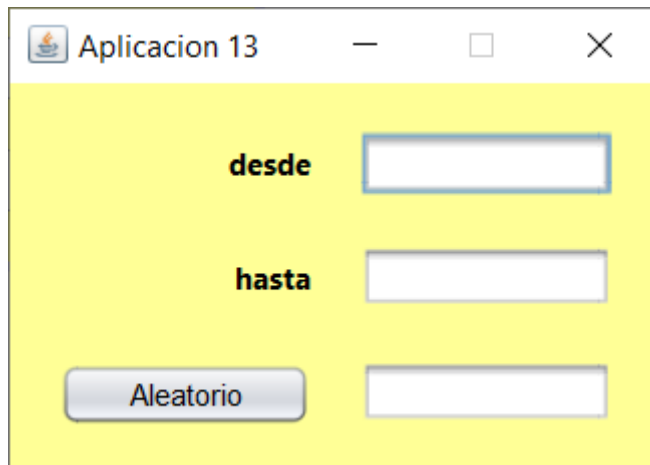
Mostrar un cuadro de dialogo de entrada y comprobar que se ha cancelado la introducción del dato

```
Object o = JOptionPane.showInputDialog(v, "Selecciona el matiz",
"MATIZ ROJO", JOptionPane.PLAIN_MESSAGE, null, valores, valores[0]);
if(o==null)
{
}
```

Formularios modales

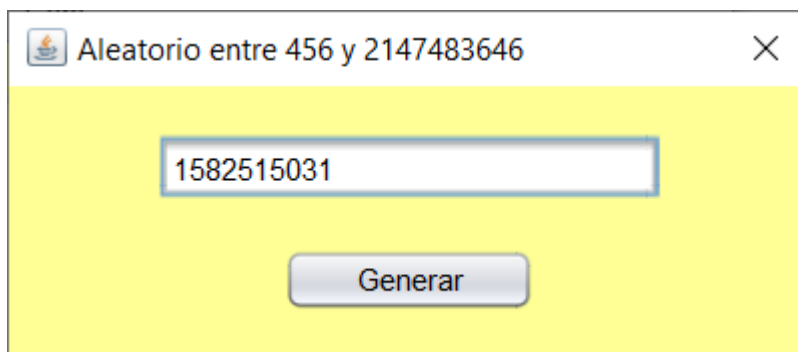
EJRCICIO (ver código para lanzar el formulario modal desde el formulario principal y definir el código de inicio del formulario modal)

32. Hágase un formulario que obtenga un número aleatorio entre dos dados. Los datos desde y hasta se introducirán en sendas cajas de texto del formulario principal. Al pulsar un botón se mostrará un formulario modal al cual se le pasa el desde y el hasta. Desde dicho formulario modal se puede generar números aleatorios al pulsar un botón. También habrá un botón para cerrar el formulario modal y pasarle al formulario principal el último de los números generados.



Formulario principal

- rr. El panel principal tendrá un tamaño de 260 x 155 y un margen de 20
 - ss. Todos los controles tendrán un tamaño de 100 x 25 y están separados entre sí 20
 - tt. Las etiquetas tendrán el texto alineado a la derecha y estará en negrita
 - uu. Las dos cajas de texto donde se introducen los datos (las dos primeras) tendrán el texto alineado a la derecha y sólo admitirán la cadena vacía o valores entre 0 y Integer.MAX_VALUE-1 (no se verificará que el valor desde es inferior al valor hasta)
 - vv. La última caja de texto será de sólo lectura con fondo blanco y con el texto alineado a la derecha
- 1) Al pulsar el botón se mostrará el formulario modal al cual se le pasa el contenido de las cajas de texto. Al cerrarse el formulario modal se mostrará en la caja de texto de sólo lectura el último número aleatorio generado.



Formulario modal

- a) El panel principal tendrá un tamaño de 320 x 110, un margen superior e inferior de 20 y un margen izquierdo y derecho de 60
 - b) La caja de texto es de sólo lectura con fondo blanco y con un tamaño de 200 x 25
 - c) El botón tiene un tamaño de 100 x 25 con un margen izquierdo de 110.
 - d) La caja de texto y el botón está separado verticalmente 20
-
- 1) Al cargarse el formulario se muestra como título el texto de “Aleatorio entre a y b”, siendo “a” el valor desde y “b” el valor hasta.
 - 2) Al cargarse el formulario se pulsa desde código el botón “generar”
 - 3) Al pulsar el botón de “generar” se genera un número aleatorio entre el desde y hasta, ambos incluidos mostrados en el título.

Lanzar desde el formulario principal el formulario modal pasando datos

En el método de “inicio” de la “Vista1” del formulario modal se le pasa la vista de la cual depende y los datos necesarios del formulario principal

```
Modelo1 m1 = Vista1.inicio(v, v.getDesde().getText(),v.getHasta().getText());
```

Se devuelve el modelo del formulario modal para acceder a los datos obtenidos en dicho formulario modal y poder pasárselos al formulario principal

Código de inicio del formulario modal

En el método de inicio del formulario modal se le pasa la vista de la cual depende y los datos necesarios. Se devuelve el modelo del formulario modal donde se almacenan los datos para ser pasados al formulario principal.

El formulario modal se ejecuta en el mismo hilo que el del formulario dependiente.

```
public static Modelo1 inicio(JFrame f, String a, String b) {  
    Vista1 v = new Vista1(f, true);  
    Modelo1 m = new Modelo1(a,b);  
    controladores.Controlador1 c = new controladores.Controlador1(m,v);  
    v.setVisible(true);  
  
    return m;  
}
```

FORMULARIOS CON ACCESO A FICHEROS

- La clase fsg.ficheros.Buffered
- La clase fsg.ficheros.Data
- La clase fsg.ficheros.Objeto
- La clase fsg.ficheros.UnObjeto

BufferedReader y BufferedWriter

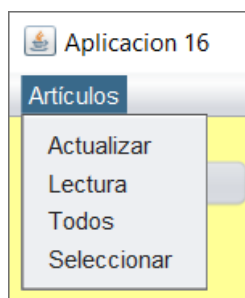
EJERCICIO (ver la clase Buffered y métodos privados de interés)

33. Se considera un fichero “fic/artículos.txt” con dos campos:

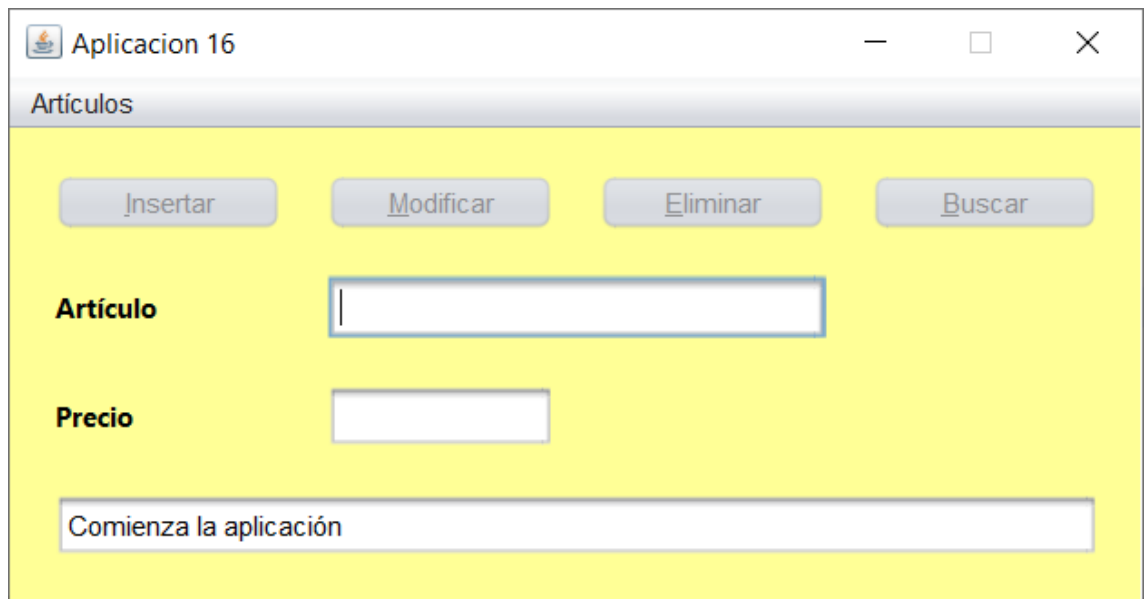
- a. Nombre del artículo → Palabra en mayúsculas sin diéresis ni acentos con un máximo de 20 caracteres. Representa el nombre del artículo. No puede haber dos artículos con el mismo nombre.
- b. Precio → Valor real entre 0 y 100.00, ambos incluidos o el valor null. Representa el valor actual del artículo, el cual puede ser desconocido.

Hágase un menú con los siguientes formularios:

- a) Actualizar → Se podrá insertar, modificar, eliminar y buscar (por el campo del nombre del artículo) registros en el fichero “fic/artículos.txt”
- b) Lectura → Se podrá consultar el primero y el último en orden alfabético por el nombre del artículo, y el inmediatamente siguiente y anterior a un nombre de artículo determinado.
- c) Todos → Se mostrarán en una tabla todos los artículos
- d) Seleccionar → Se mostrarán sólo los artículos que comienza por un determinado prefijo y cuyo precio está entre dos cantidades determinadas. Puede ser que no se tenga que filtrar por prefijo, precio mínimo y/o precio máximo.



Formulario Actualizar

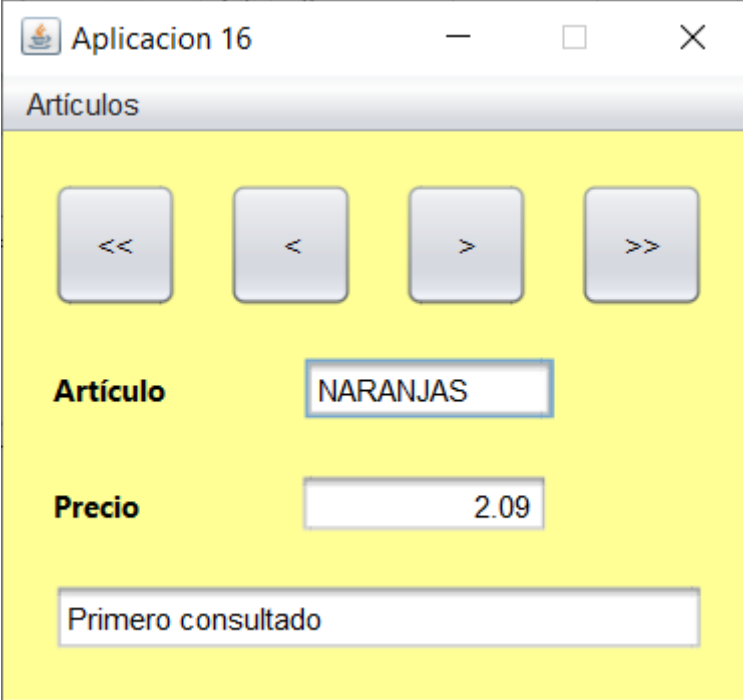


- a) El panel principal tiene un tamaño de 500 x 200 con un relleno de 20
 - b) Los botones y las etiquetas tienen un tamaño de 100 x 25. Los botones están deshabilitados, no pueden recibir el foco con el tabulador y con teclas de acceso directo. Las etiquetas están en negrita.
 - c) La caja de texto donde se introduce el nombre del artículo tiene un tamaño de 220 x 25. En dicha caja sólo se pueden introducir palabras en mayúsculas sin acentos ni diéresis con un máximo de 20 letras.
 - d) La caja donde se introduce el precio tiene un tamaño de 100 x 25, el texto alineado a la derecha y sólo admite valores entre 0.00 y 100.00, ambos inclusive. Se dirá que el precio no está completo si de aparecer el punto decimal, este fuera el último carácter.
 - e) La caja de texto de estado es de sólo lectura con fondo blanco y tiene un tamaño de 460 x 25
- 1) Los cuatro botones estarán deshabilitados si el nombre del artículo es la cadena vacía. Además, de ser el nombre del artículo distinto de la cadena vacía, los botones “Insertar” y “Modificar” si el precio del artículo está incompleto.
 - 2) Al pulsar el botón de “insertar”, se insertará el nuevo artículo siempre y cuando no exista ya el artículo. De existir, se pondrá en la caja de estado el mensaje “Ya existe el artículo” y si no, se pondrá “Artículo insertado”
 - 3) Al pulsar el botón de “buscar”, se buscará el artículo cuyo nombre se ha introducido. De no existir, se vaciará la caja de texto del precio y se pondrá como mensaje “No existe el artículo”; de existir, se muestra el precio y se pone como mensaje “Artículo encontrado”.
 - 4) Al pulsar el botón de “eliminar” se elimina el artículo cuyo nombre se ha introducido. Si no existe el artículo, se vacían la caja de texto del precio y se pone como mensaje “No existe el artículo a eliminar”; y de existir, se muestra el precio del artículo eliminado y se pone como mensaje “Artículo eliminado”.
 - 5) Al pulsar el botón de modificar, se modifica el precio del artículo introducido. De no existir el artículo a modificar, se pone como mensaje “No existe el

artículo a modificar”; de existir, se modifica el artículo y se pone como mensaje “Artículo modificado”

- 6) Al modificarse el nombre del artículo se pone como mensaje “Nombre del artículo modificado”
- 7) Al modificarse el precio del artículo se pone como mensaje “Precio modificado”

Formulario Lectura

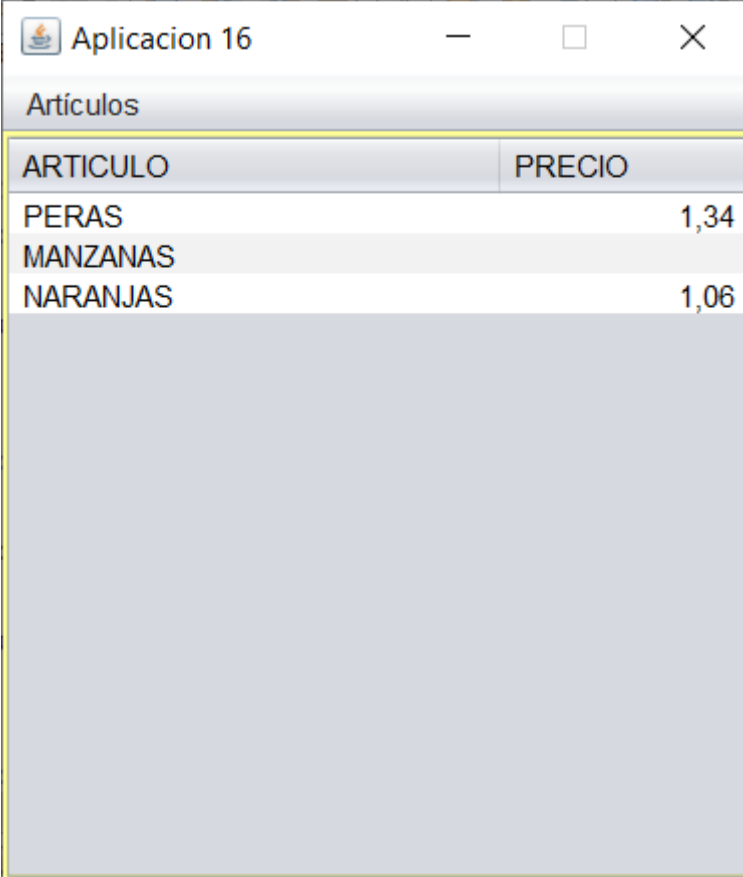


- a) El panel principal tiene un tamaño de 300 x 220 con un relleno de 20
 - b) Los botones tienen un tamaño de 50 x 50 y no pueden recibir el foco con el tabulador.
 - c) Las etiquetas y las dos primeras cajas de texto tienen un tamaño de 100 x 25. Las etiquetas tienen el texto en negrita. La caja de texto del precio es de solo lectura con fondo blanco y texto alineado a la derecha.
 - d) La caja de texto de “estado” es de solo lectura, con fondo blanco y con un tamaño de 260 x 25.
- 1) Al pulsar el botón del primero (<<) se obtiene el primer artículo (el menor en orden alfabético). Si no hubiera artículos, se vacían las cajas de texto y se pone como mensaje “No hay datos”; de haber artículos, se muestra el primer artículo y se pone como mensaje “Primero consultado”.
 - 2) Al pulsar el botón del último (>>) se obtiene el último artículo (el mayor en orden alfabético). Si no hubiera artículos, se vacían las cajas de texto y se pone como mensaje “No hay datos”; de haber artículos, se muestra el último artículo y se pone como mensaje “Último consultado”.
 - 3) Al pulsar el botón de siguiente (<) se obtiene el artículo inmediatamente siguiente al introducido. Si no hubiera siguiente, se vacía la caja de texto del

precio y se pone como mensaje “No hay siguiente”; si no, se muestra el siguiente artículo y se pone como mensaje “Siguiente consultado”.

- 4) Al pulsar el botón de anterior (>) se obtiene el artículo inmediatamente anterior al introducido. Si no hubiera anterior, se vacía la caja de texto del precio y se pone como mensaje “No hay anterior”; si no, se muestra el anterior artículo y se pone como mensaje “Anterior consultado”.

Formulario Todos



ARTICULO	PRECIO
PERAS	1,34
MANZANAS	
NARANJAS	1,06

- a) El panel principal tiene un tamaño de 300 x 300 al igual que el panel contenedor de la tabla.
 - b) La primera columna de la tabla tiene un ancho de 200 y la segunda, de 100
 - c) No se puede cambiar la anchura de las columnas ni reordenar.
 - d) Se puede ordenar las filas por los valores de sus columnas.
- 1) Al cargarse el formulario se mostrarán todos los artículos.

Formulario Seleccionar

The screenshot shows a window titled 'Aplicacion 16' with a yellow background. At the top, there's a title bar with standard window controls. Below it, a subtitle 'Articulos' is centered. The main area contains three labels on the left: 'Artículo', 'Mínimo', and 'Máximo', each followed by a text input box. To the right of these inputs is a table with two columns: 'ARTICULO' and 'PRECIO'. The table contains three rows of data: 'PERAS' with price '1,34', 'MANZANAS', and 'NARANJAS' with price '1,06'. The table has a light gray header and alternating row colors.

ARTICULO	PRECIO
PERAS	1,34
MANZANAS	
NARANJAS	1,06

- a) El panel principal tiene un tamaño de 560 x 340 con un relleno de 20
 - b) Las etiquetas y cajas de texto tienen un tamaño de 100 x 25
 - c) El panel que contiene la tabla tiene un tamaño de 300 x 300
 - d) Las etiquetas tienen el texto en negrita
 - e) En la primera caja de texto sólo se pueden introducir palabras en mayúsculas sin acentos ni diéresis con un máximo de 20 letras.
 - f) En la segunda y tercera caja de texto sólo se puede introducir la cadena vacía y valores reales entre 0.00 y 100.00, ambos inclusive. El texto de tales cajas estará alineado a la derecha.
 - g) La primera columna de la tabla tiene un ancho de 200 y la segunda, de 100. No se puede cambiar la anchura de las columnas ni reordenar. Se puede ordenar las filas por los valores de sus columnas.
 - h) Todos los controles están separados horizontal y verticalmente entre sí 20.
- 1) Al cargarse inicialmente se muestran en la tabla todos los artículos del fichero.
 - 2) Al modificarse el contenido de cualquiera de las cajas de texto, se rellenará la tabla con los artículos que cumplen las siguientes condiciones:
 - La primera caja contiene la cadena vacía o el nombre del articulo comienza por el valor almacenado en la primera caja
 - La segunda caja contiene la cadena vacía o el precio del artículo, de ser distinto de null, es mayor o igual que el precio almacenado en la segunda caja.
 - La tercera caja contiene la cadena vacía o el precio del artículo, de ser distinto de null, es menor o igual que el precio almacenado en la tercera caja.

```

/* METODOS PRIVADOS */
private Object[] buscarArticulo(String s) throws IOException, ClassNotFoundException
{
    Object[] enc = null;
    bfArticulos.abrir();
    while((enc=bfArticulos.leer())!=null)
    {
        String cad = (String) enc[0];
        if(cad.equals(s)) break;
    }

    return enc;
}

private void copiar(Buffered desde, Buffered a) throws IOException, ClassNotFoundException
{
    Object[] dat = null;

    a.cerrar();
    a.vaciar();

    desde.abrir();
    while((dat=desde.leer())!=null)
    {
        String datArt = (String) dat[0];
        Double datPre = (Double) dat[1];
        a.escribir(datArt,datPre);
    }
    desde.cerrar();
    desde.eliminar();
}

```

DataInputStream y DataOutputStream

EJERCICIO (ver la clase Data y métodos privados de interés)

- 34.** Resuélvase el ejercicio anterior almacenando los datos en un fichero binario de tipos básicos de nombre “fic/artículos.bin”. El valor null para el precio se representa con el valor -1.0. (Aplicación 18)

```

/* METODOS PRIVADOS */
Son los mismos que en el ejercicio anterior cambiando Buffered por Data.

```

ObjectInputStream y ObjectOutputStream

EJERCICIO (ver la clase Objeto, métodos privados de interés y la clase auxiliar Artículo)

- 35.** Resuélvase el ejercicio anterior almacenando los datos en un fichero binario de objetos de nombre “fic/artículos.obj”. (Aplicación 36)


```

/* CLASE ARTICULO */
package clases;

import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.Serializable;
import java.util.Objects;

public class Articulo implements Comparable<Articulo>, Serializable{

    private String nombre;
    private Double precio;

    //<editor-fold defaultstate="collapsed" desc="metodos get y set">
    public static final String PROP_PRECIO = "precio";

    public Double getPrecio() {
        return precio;
    }

    public void setPrecio(Double precio) {
        if(precio!=null && (precio<0 || precio>100 || (int)(precio*100)/100.0!=precio))
            throw new PrecioIncorrectoException("Precio incorrecto");

        Double oldPrecio = this.precio;
        this.precio = precio;
        propertyChangeSupport.firePropertyChange(PROP_PRECIO, oldPrecio, precio);
    }

    public static final String PROP_NOMBRE = "nombre";

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        if(!nombre.matches("[A-ZÑ]{0,20}"))
            throw new ArticuloIncorrectoException("Articulo incorrecto");

        String oldArticulo = this.nombre;
        this.nombre = nombre;
        propertyChangeSupport.firePropertyChange(PROP_NOMBRE, oldArticulo, nombre);
    }

    private transient final PropertyChangeSupport propertyChangeSupport = new
PropertyChangeSupport(this);

    public void addPropertyChangeListener(String propiedad,PropertyChangeListener listener) {
        propertyChangeSupport.addPropertyChangeListener(propiedad,listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertyChangeSupport.removePropertyChangeListener(listener);
    }

```

```

}
//</editor-fold>

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Artículo other = (Artículo) obj;
    return Objects.equals(this.nombre, other.nombre);
}

@Override
public int compareTo(Artículo o) {
    return nombre.compareTo(o.nombre);
}

public Artículo(String nombre, Double precio) {
    setNombre(nombre);
    setPrecio(precio);
}

public Artículo()
{

}

public static class PrecioIncorrectoException extends RuntimeException {

    public PrecioIncorrectoException(String mensaje) {
        super(mensaje);
    }
}

public static class ArtículoIncorrectoException extends RuntimeException {

    public ArtículoIncorrectoException(String mensaje) {
        super(mensaje);
    }
}

}

/* METODOS PRIVADOS */
private Artículo buscarArtículo(String s) throws IOException, ClassNotFoundException
{
    Artículo enc = null;
    obArticulos.abrir();

```

```

while((enc=objArticulos.leer())!=null)
{
    String cad = enc.getNombre();
    if(cad.equals(s)) break;
}

return enc;
}

private void copiar(Objeto<Articulo> desde, Objeto<Articulo> a) throws IOException,
ClassNotFoundException
{
    Articulo dat = null;

    a.cerrar();
    a.vaciar();

    desde.abrir();
    while((dat=desde.leer())!=null)
        a.escribir(dat);

    desde.cerrar();
    desde.eliminar();
}

```

La clase UnObjeto

EJERCICIO (ver la clase UnObjeto)

36. Resuélvase la aplicación 34 de forma que al cargar el formulario se rellenará la tabla con los datos almacenados en un fichero “fic/modeloTabla.uob”. Al cerrarse el formulario se volcarán los datos actuales de la tabla en dicho fichero.

The screenshot shows a Java Swing window titled "Aplicacion 39". The window has a yellow background. At the top, there are three buttons: "Insertar", "Eliminar", and "Modificar". Below these buttons, there are two input fields. The first is labeled "Nombre" and the second is labeled "Edad". At the bottom, there is a text area containing the text "Comienza la aplicación". On the right side of the window, there is a table with two columns: "Nombre" and "Edad". The table contains two rows of data: "Pedro" with "11" and "Antonio" with "10".

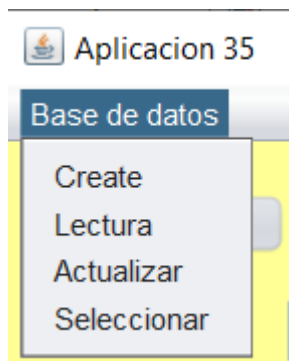
Nombre	Edad
Pedro	11
Antonio	10

FORMULARIOS CON ACCESO A BASE DE DATOS

- La clase fsg.bd y la librería mysql.jar
- Actualizaciones sin parámetros
- Lecturas con y sin parámetros
- Actualizaciones con parámetros

Acceso a base de datos

37. Hágase un menú de formularios para poder crear base de datos y tablas, consultar secuencialmente una tabla, seleccionar filas de una tabla y actualizar una tabla



Formulario create (Actualización sin parámetros)

Hágase un formulario que permita con un menú de botones crear y eliminar la base de datos JAVA, las tablas EQUIPOS y JUGADORES de dicha base de datos, e insertar y borrar registros en tales tablas. Habrá una caja de texto de solo lectura en la que se indicará el resultado de cada consulta o el error cometido; en el caso de la inserción y eliminación de filas se indicará el número de filas insertadas o eliminadas en cada una de las tablas.



- a) El panel principal tiene un tamaño de 420 x 275 y tiene un relleno de 20
- b) Todos los botones tienen un tamaño de 180 x 50
- c) La caja de texto tiene un tamaño de 380 x 25
- d) Todos los controles están separados entre sí y del panel principal 20

1) Al pulsar el primer botón se ejecuta el siguiente comando

```
"CREATE DATABASE IF NOT EXISTS java DEFAULT CHARACTER SET utf8 COLLATE
utf8_spanish_ci"
```

2) Al pulsar el segundo botón se ejecuta el comando

```
"DROP DATABASE IF EXISTS java;"
```

3) Al pulsar el tercer botón se ejecutan los comandos

```
"CREATE TABLE IF NOT EXISTS EQUIPOS( "+
    "ID INT(5) UNSIGNED NOT NULL,"+
    "NOMBRE VARCHAR(40) NOT NULL,"+
    "ESTADIO VARCHAR(40) NOT NULL,"+
    "POBLACION VARCHAR(20) NOT NULL,"+
    "PROVINCIA VARCHAR(20) NOT NULL,"+
    "PRIMARY KEY (ID), "+
    "UNIQUE KEY UK_nombre (NOMBRE), "+
    "UNIQUE KEY UK_estadio (ESTADIO)) ENGINE = InnoDB;";
```

```
"CREATE TABLE IF NOT EXISTS JUGADORES( "+
```

```
"ID INT(5) UNSIGNED NOT NULL,"+
"EQUIPO INT(5) UNSIGNED NOT NULL,"+
"NOMBRE VARCHAR(40) NOT NULL,"+
"DORSAL INT(2) UNSIGNED NULL,"+
"EDAD INT(3) UNSIGNED NULL,"+
"PRIMARY KEY (ID)," +
"UNIQUE KEY UK_nombre (NOMBRE), "+
"FOREIGN KEY (EQUIPO) REFERENCES EQUIPOS (ID) "+
"ON DELETE RESTRICT ON UPDATE CASCADE) ENGINE = InnoDB;";
```

4) Al pulsar el cuarto botón se ejecutan los comandos

```
"DROP TABLE IF EXISTS JUGADORES;"
```

```
"DROP TABLE IF EXISTS EQUIPOS;"
```

5) Al pulsar el quinto botón se ejecutan los comandos

```
"INSERT IGNORE INTO EQUIPOS VALUES"+
```

```
"(1,\"ESTEPONA\", \"MONTERROSO\", \"ESTEPONA\", \"MALAGA\"),"+
"(2,\"ALCORCON\", \"SANTO DOMINGO\", \"ALCORCON\", \"MADRID\"),"+
"(3,\"PORCUNA\", \"SANCRISTOBAL\", \"PORCUNA\", \"JAEN\");"
```

```
"INSERT IGNORE INTO JUGADORES VALUES"+
```

```
"(1,1,\"JOSE ANTONIO\",1,42),"+
"(2,1,\"IGNACIO\",2,62),"+
"(3,1,\"DIEGO\",3,20),"+
"(4,2,\"TURRION\",1,37),"+
"(5,2,\"LUIS ABEL\",2,NULL),"+
"(6,2,\"ISAAC\",3,40),"+
"(7,3,\"JUAN FRANCISCO\",NULL,33),"+
"(8,3,\"PARRA\",2,37),"+
"(9,3,\"RAUL\",NULL,NULL);"
```

6) Al pulsar el sexto botón se ejecutan los comandos

```
"DELETE FROM JUGADORES;"
```

```
"DELETE FROM EQUIPOS;"
```

Formulario lectura (Consulta con una sola fila)

Hágase un formulario que permita consultar secuencialmente la tabla JUGADORES de la base de datos JAVA

Aplicacion 35

Base de datos

<< Código 1

< Nombre JOSE ANTONIO

> Dorsal 1

>> Edad 42

Equipo 1

Primero consultado

- El panel principal tiene un tamaño de 350 x 340.
- Todos los botones tienen un tamaño de 50 x 50
- Todas las etiquetas tienen un tamaño de 70 x 25
- La primera caja de texto tiene un tamaño de 80 x 25; la segunda, de 150 x 25; la tercera, cuarta y quinta, de 40 x 25 y la última de 310 x 25.
- Todas las cajas de texto son de solo lectura salvo la del código.
- Sólo la caja de texto donde se introduce el código puede recibir el foco con el tabulador.
- En la caja de texto “código” sólo se permite escribir la cadena vacía o números entre 0 y 99999, ambos inclusive.
- Al realizar una consulta los valores que tengan el valor NULL (las columnas “edad” y “dorsal” son opcionales) se mostrarán en su correspondiente caja de texto con la cadena vacía.
- Los botones “<” (anterior) y “>” (siguiente) sólo se habilitarán cuando el contenido de la caja de texto “código” sea distinto de la cadena vacía.
- Si cualquiera de las consultas (primero, anterior, siguiente y ultimo) fueran exitosa, se mostrará en la caja de texto inferior los siguientes mensajes: “Primero consultado”, “Siguiente consultado”, “Anterior consultado” y “Último consultado”.

- k) Si al consultar el primero (“<<”) o el último (“>>”) la consulta no tuviera éxito (la tabla está vacía) se pondrá en la caja inferior el mensaje “No hay datos” y se vaciarán el resto de las cajas de texto.
- l) Si al consultar el anterior (“<”) o el siguiente (“>”) la consulta no tuviera éxito (se intenta consulta una fila anterior a la primera o posterior a la última) se pondrá en la caja inferior el mensaje “No hay siguiente” o “No hay anterior” y se vaciarán todas las cajas de texto salvo la del código.

Primero

```
"SELECT id, nombre, edad, dorsal, equipo FROM JUGADORES  
ORDER BY ID LIMIT 1"
```

Último

```
"SELECT id, nombre, edad, dorsal, equipo FROM JUGADORES  
ORDER BY ID DESC LIMIT 1"
```

Siguiente

```
"SELECT id, nombre, edad, dorsal, equipo FROM JUGADORES  
WHERE id > ? ORDER BY ID LIMIT 1"
```

Anterior

```
"SELECT id, nombre, edad, dorsal, equipo FROM JUGADORES  
WHERE id < ? ORDER BY ID DESC LIMIT 1"
```

Formulario seleccionar (Consulta con múltiples filas)

Hágase un formulario que muestre en una tabla todos los jugadores de todos los equipos o de un determinado equipo cuya edad no cumpla ninguna restricción o sea menor y/o mayor que una determinada.

Aplicacion 35

Base de datos

	Id	Nombre	Edad	Dorsal	Equipo
Equipo	1	JOSE ANTONIO	42	1	1
<input type="text"/>	2	IGNACIO	62	2	1
	3	DIEGO	20	3	1
	4	TURRION	37	1	2
	5	LUIS ABEL		2	2
Edad mínima	6	ISAAC	40	3	2
<input type="text"/>	7	JUAN FRANCISCO	33		3
	8	PARRA	37	2	3
Edad máxima	9	RAUL			3
<input type="text"/>					

- El panel principal tiene un tamaño de 520 x 290
 - El panel que contiene la tabla tiene un tamaño de 400 x 290
 - Todas las etiquetas y cajas de texto tienen un tamaño de 60 x 25 y con una separación vertical entre sí de 20 y con un margen superior, inferior, izquierdo y derecho de 20.
 - En la primera caja de texto sólo se podrán introducir valores entre 0 y 99999, ambos inclusive.
 - En la segunda y tercera caja de texto sólo se podrá introducir la cadena vacía o un número entre 0 y 120, ambos inclusive.
 - La primera, tercera y cuarta columna tendrán una anchura de 50; la segunda de 160 y la última de 90. Las columnas no se podrán redimensionar ni mover. Las filas no se podrán seleccionar. Se podrá ordenar las filas por cada una de las columnas.
- Inicialmente se muestra en la tabla todos los jugadores.
 - Al modificarse cualquiera de las cajas de texto se obtendrá la tabla que cumple tales restricciones.

Formulario de actualización (actualizaciones con parámetros)

Hágase un formulario que permita insertar, modificar, eliminar y buscar una fila en la tabla de JUGADORES.

Aplicacion 35

Base de datos

Insertar Modificar Eliminar Buscar

Código

Nombre

Dorsal

Edad

Equipo

Comienza la aplicación

Formulario en diseño con etiquetas que se harán invisibles:

Insertar Modificar Eliminar Buscar

Código *

Nombre *

Dorsal *

Edad *

Equipo *

- a) El panel principal tendrá un tamaño de 460 x 335 con un relleno de 20
 - b) Todos los botones tienen un tamaño de 90 x 25 y tienen teclas de acceso directo y no pueden recibir el foco con el tabulador.
 - c) Todas las etiquetas que describen el contenido de las cajas de texto tienen un tamaño de 70 x 25 y el texto está en negrita
 - d) Las cajas de texto donde se introducen datos tienen un tamaño de 80 x 25, salvo la segunda que tiene un tamaño de 305 x 25. Todas tienen el texto alineado a la derecha.
 - e) Las etiquetas con el “*” tienen un tamaño de 25 x 25 y el texto está en negrita de color rojo y centrado horizontalmente.
 - f) La última caja es de solo lectura y tiene un tamaño de 420 x 25
 - g) Todos los controles están separados horizontal y verticalmente 20
 - h) En las cajas de texto “código” y “equipo” sólo se admiten valores entre 0 y 99999, ambos inclusive.
 - i) En la caja de texto “nombre” sólo se admiten palabras en español sin acentos ni diéresis separadas por un espacio y con un máximo de 40 caracteres.
 - j) En las cajas de texto “edad” sólo se admiten valores entre 0 y 120, ambos inclusive.
 - k) En la caja de texto “dorsal” sólo se admiten valores entre 0 y 99, ambos inclusive.
 - l) En todas las cajas de texto se admite la cadena vacía.
- 1) Los botones “buscar” y “eliminar” sólo se habilitarán cuando la caja de texto de “código” sea distinto de la cadena vacía.
 - 2) Los botones “insertar” y “modificar” se habilitarán cuando las cajas de texto “código”, “nombre” y “equipo” sean distintas de la cadena vacía.

- 3) Las cajas de texto se pondrán con fondo amarillo cuando reciban el foco y con fondo blanco cuando lo pierdan.
- 4) Cada vez que se modifique el contenido de una caja de texto, se indicará en la barra de estado que se ha modificado el contenido de dicho control (código, nombre, edad, dorsal o equipo) y se quitarán los posibles mensajes de error (mostrados en los cartelitos de las cajas de texto con “*”) que hubieran.
- 5) Al pulsar cualquiera de los botones lo primero que se hace es quitar todos los posibles mensajes de error (mostrados en los cartelitos de las cajas de texto con “*”) que hubieran.
- 6) Al pulsar el botón de “buscar” se buscará el jugador de id introducido en la caja de texto “código”. De existir, se muestran los datos y se pone en la barra de estado “Jugador buscado”; de no existir, se vacían todas las cajas de texto menos la del código, se activa un mensaje de error asociado a la caja de texto “código” con el mensaje “No existe el jugador a buscar” y se pone en la barra de estado el mensaje “Errores al buscar un jugador”.

Aplicacion 35

Base de datos

Insertar Modificar Eliminar Buscar

Código 34 *

Nombre No existe el jugador a buscar

Dorsal

Edad

Equipo

Errores al buscar jugador

- 7) Al pulsar el botón de “eliminar” se eliminará el jugador de id introducido en la caja de texto “código”. De existir, se elimina dicho jugador y se muestran los datos del jugador eliminado y se pone en la barra de estado “Jugador eliminado”; de no existir, se vacían todas las cajas de texto menos la del código, se activa un mensaje de error asociado a la caja de texto “código” con el mensaje “No existe el jugador

a eliminar” y se pone en la barra de estado el mensaje “Errores al eliminar un jugador”.

Aplicacion 35

Base de datos

Insertar Modificar Eliminar Buscar

Código 34 *

No existe el jugador a eliminar

Nombre

Dorsal

Edad

Equipo

Errores al eliminar jugador

- 8) Al pulsar el botón de “insertar” se comprueba si ya existe el código de jugador a insertar, el nombre del jugador ya está repetido, el equipo existe, de existir el equipo y el dorsal sea conocido se comprueba que no está repetido en dicho equipo y que la edad es un valor entre 16 y 50, ambos sin incluir. De haber errores se activará en la caja correspondiente mensajes de error del tipo “Ya existe el jugador”, “El nombre ya existe”, “El equipo no existe”, “El dorsal ya existe en ese equipo” y “La edad tiene que ser mayor que 16 y menor que 50”. Además de haber errores, se pondrá en la barra de estado el mensaje “Errores al insertar”. De no haber errores, se inserta el nuevo jugador y se pone en la barra de estado “Jugador insertado”.
- 9) Al pulsar el botón de “modificar” se comprueba si no existe el código de jugador a modificar, si el nombre del jugador ya está repetido en otro jugador distinto del que se está modificando, el equipo existe, de existir el equipo y el dorsal sea conocido se comprueba que no está repetido en dicho equipo para otro jugador distinto del que se está modificando y que la edad es un valor entre 16 y 50, ambos sin incluir. De haber errores se activará en la caja correspondiente mensajes de error del tipo “No existe el jugador”, “El nombre ya existe”, “El equipo no existe”, “El dorsal ya existe en ese equipo” y “La edad tiene que ser mayor que 16 y menor

que 50". Además de haber errores, se pondrá en la barra de estado el mensaje "Errores al modificar". De no haber errores, se modificarán los datos del jugador jugador y se pone en la barra de estado "Jugador modificado".

Consultas necesarias

```
"SELECT * FROM JUGADORES WHERE ID = ?;"
```

```
"SELECT * FROM JUGADORES WHERE NOMBRE = ?;"
```

```
"SELECT * FROM EQUIPOS WHERE ID = ?;"
```

```
"SELECT * FROM JUGADORES WHERE EQUIPO = ? AND DORSAL = ?;"
```

```
"SELECT * FROM JUGADORES WHERE ID <> ? AND NOMBRE = ?;"
```

```
"SELECT * FROM JUGADORES WHERE ID <> ? AND EQUIPO = ? AND DORSAL = ?;"
```

```
"SELECT id, nombre, edad, dorsal, equipo FROM JUGADORES WHERE ID = ?"
```

```
"INSERT INTO JUGADORES VALUES (?, ?, ?, ?, ?);" "
```

```
"DELETE FROM JUGADORES WHERE ID = ?"
```

```
"UPDATE JUGADORES SET NOMBRE = ?, DORSAL = ?, EDAD = ?, EQUIPO = ?  
WHERE ID = ?;"
```