Capítulo 8

Interfaces gráficas de usuario

8.1. Resultados de aprendizaje

5. Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

b) Se han aplicado formatos en la visualización de la información.

c) Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.

d) Se han utilizado ficheros para almacenar y recuperar información.

e) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

g) Se han programado controladores de eventos.

h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

6. Escribe programas que manipulen información, seleccionando y utilizando tipos avanzados de datos.

g) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.

8.2. Introducción

Hasta ahora en la mayoría de los ejercicios hemos utilizado la salida estándar (System.out.print) y las ventanas predefinidas (JOptionPane.showInputDialog, JOptionPane.showMessageDialog) para interactuar con el usuario. Esto nos ha permitido centrarnos en la programación orientada a objetos y la gestión de errores, sin tratar a la vez con botones, ventanas y demás elementos gráficos. Sin embargo, hoy en día, cualquier proyecto dispone de un entorno gráfico más rico (páginas web, ventanas, etc..) para interactuar con el usuario.

Para poder crear entornos gráficos en java es necesario importar dos librerías las cuales dotan de varios componentes y soporte para gestionar la interacción con el usuario. Las dos librerías son:

```
1 java.awt.*;
2 javax.swing.*;
```

La librería **java.awt** es una librería del modo gráfico que depende directamente del sistema operativo.

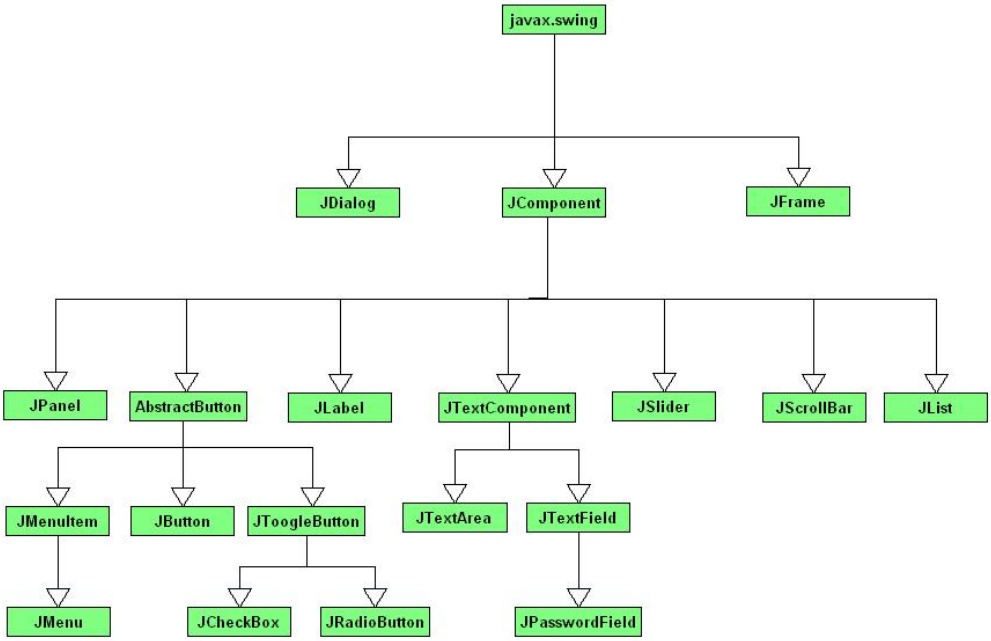
La estructura de la versión actual del AWT se puede resumir en los puntos que se exponen a continuación:

- Los **Contenedores** contienen **Componentes**, que son los controles básicos.
- No se usan posiciones fijas de los Componentes, sino que están situados a través de una disposición controlada (layouts).
- El común denominador de más bajo nivel se acerca al teclado, ratón y manejo de **eventos**.
- Alto nivel de abstracción respecto al entorno de ventanas en que se ejecute la aplicación.
- La arquitectura de la aplicación es dependiente del entorno de ventanas, en vez de tener un tamaño fijo.
- Es bastante dependiente de la máquina en que se ejecuta la aplicación (no se puede asumir que un cuadro de diálogo tendrá el mismo tamaño en cada máquina).

- Carece de un formato de recursos. No se puede separar el código de lo que es propiamente interface.

La librería **javax.swing** es una librería más estándar ya que no depende del sistema operativo, es decir, que permite una interfaz a cada sistema operativo sin cambio de código.

https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

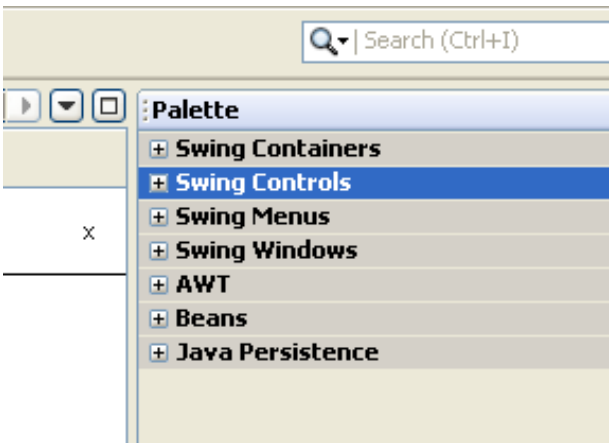


Al introducir entorno gráfico también debemos tener en cuenta que a partir de ahora vamos a desarrollar una **programación dirigida por eventos** que es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van a estar determinados por los sucesos que ocurran en el sistema o que ellos mismos provoquen.

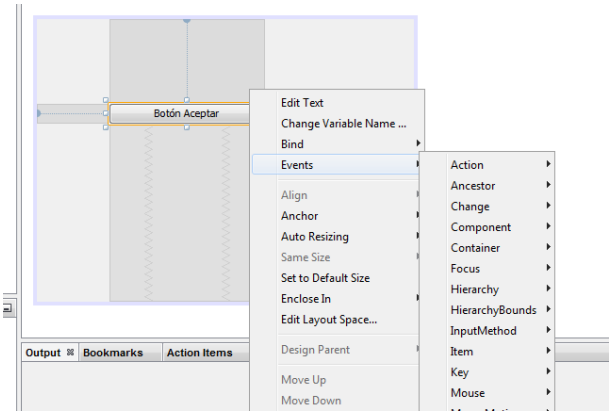
Para entender la programación dirigida por eventos, podemos oponerla a lo que no es: mientras en la programación secuencial (o estructurada) es el programador el que define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario –o lo que sea que esté accionando el programa– el que dirija el flujo del programa. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no

en cualquier momento como puede ser en el caso de la programación dirigida por eventos.

Para construir una interface gráfica de usuario hace falta:



- Un "contenedor" o **container** (JFrame o JDialog), es la ventana o parte de la ventana donde se situarán los componentes (botones, barras de desplazamiento, etc.).
- Los componentes o **controles**: botones de comando, barras de desplazamiento, cajas y áreas de texto, botones de opción y selección, etc... (JButton, JTextField, JLabel,...).
- El modelo de **eventos**. El usuario controla la aplicación actuando sobre los componentes con el ratón o con el teclado. Cada vez que el usuario realiza una determinada acción, se produce el evento correspondiente, que el sistema operativo transmite al AWT o al SWING.



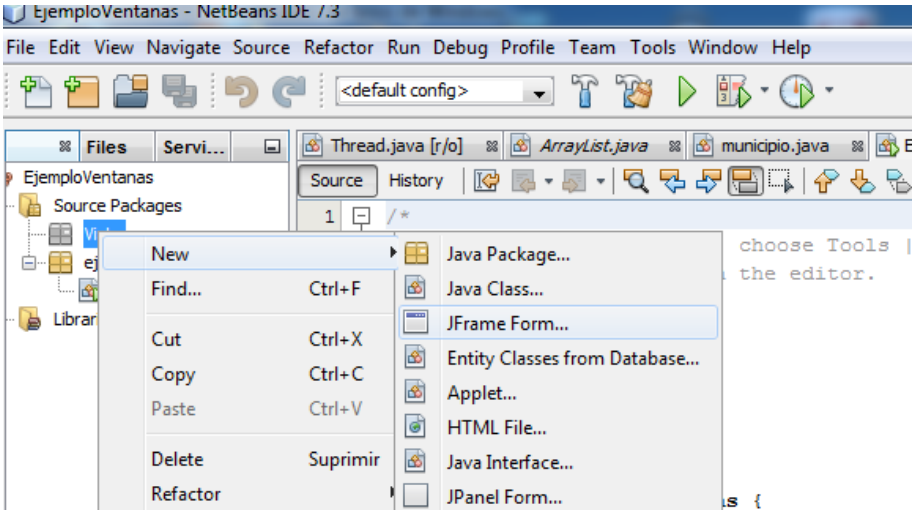
Tanto los contenedores como los componentes reciben eventos.

8.3. Proceso a seguir para crear una aplicación interactiva (orientada a eventos)

Se resumen a continuación los pasos que se pueden seguir para construir una aplicación orientada a eventos con interface gráfica de usuario:

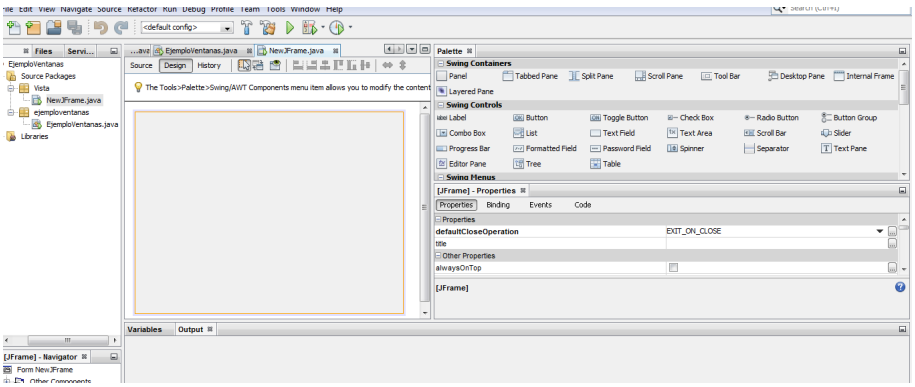
- Crear un proyecto.
  Cuando creemos un nuevo proyecto podremos dejar que nos cree la clase principal como hasta ahora o no ya que las ventanas también disponen de método main.
- Dentro del proyecto crear una clase Ventana (New JFrame Form)

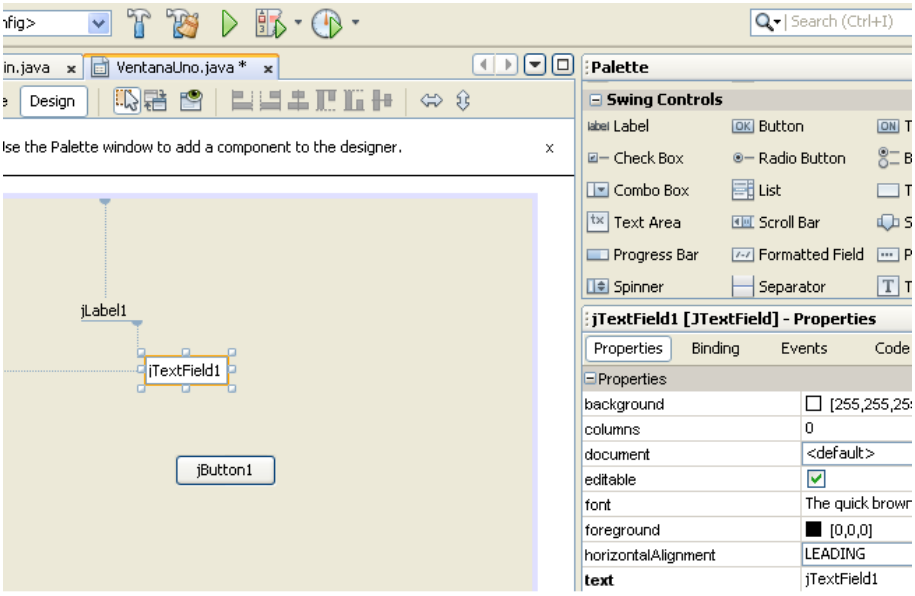
```
1  JFrame ventanaM = new JFrame(titulo);
2
3  ventanaM.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//finaliza
       el programa cuando se da click en la X
4  ventanaM.setSize(290, 150);//configurando tamaño de la ventana
5  ventanaM.setVisible(true);//configurando visualización de la
       ventana
```



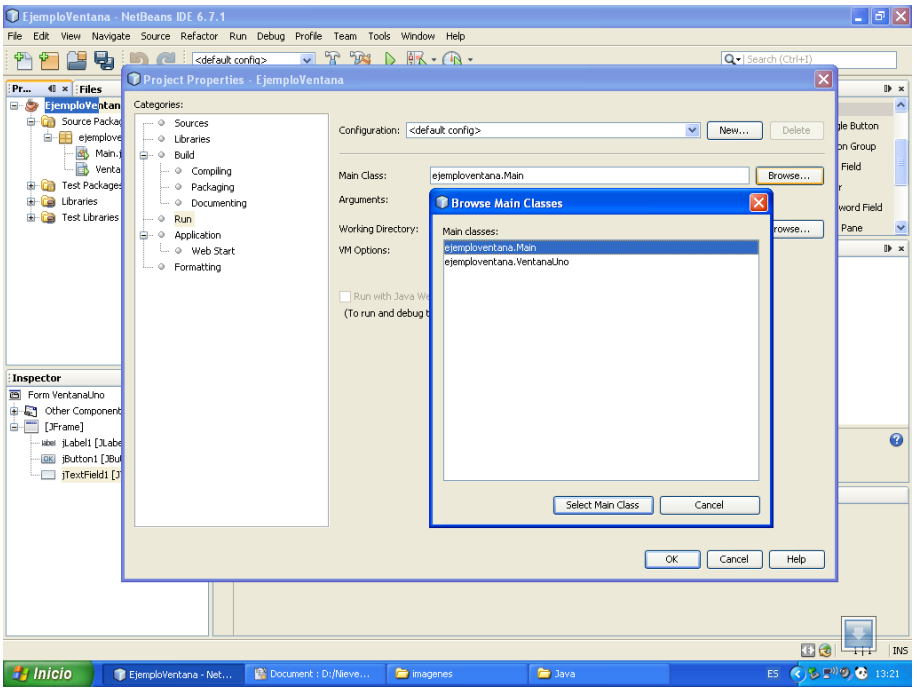
- Modificamos las propiedades que consideremos oportunas y añadimos contenedores y controles.
- El método main() deberá crear un objeto de la clase Ventana (en el que se van a introducir las componentes seleccionadas) y mostrarla por pantalla.

```
1          VentanaUno vc = new VentanaUno();
2          vc.setVisible(true);
```




Search (Ctrl+I)
VentanaUno.java *
Design
lse the Palette window to add a component to the designer.
jLabel1
jTextField1
jButton1
Palette
Swing Controls
Label
Button
Check Box
Radio Button
Combo Box
List
Text Area
Scroll Bar
Progress Bar
Formatted Field
Spinner
Separator
jTextField1 [JTextField] - Properties
Properties
Binding
Events
Code
background
columns
document
editable
font
foreground
horizontalAlignment
text
<default>
[0,0,0]
LEADING
jTextField1

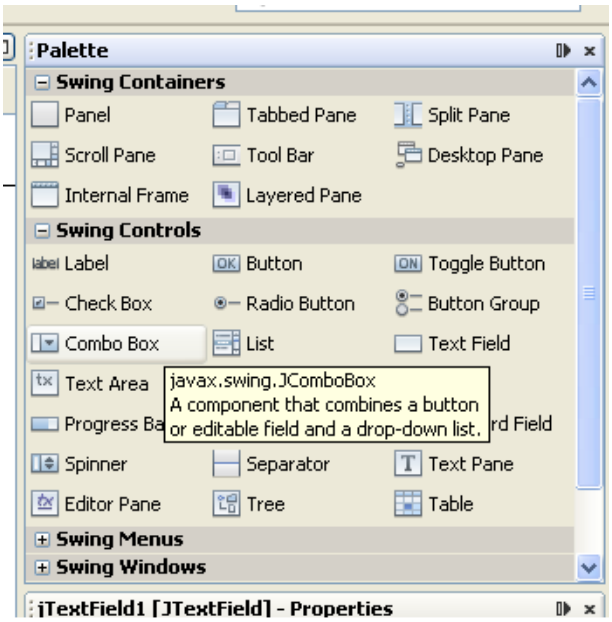
Nota:También se puede, a través de propiedades de proyecto, establecer que se inicie la ejecución por el método main asociado a la ventana.




```
1  package ejemploventana;
2
3  public class VentanaUno extends javax.swing.JFrame {
4
5      /** Creates new form VentanaUno */
6      public VentanaUno() {
7          initComponents();
8      }
9
10     @SuppressWarnings("unchecked")
11     // <editor-fold defaultstate="collapsed" desc="Generated Code">
12     private void initComponents() {
13
14         jLabel1 = new javax.swing.JLabel();
15         jButton1 = new javax.swing.JButton();
16         jTextField1 = new javax.swing.JTextField();
17
18         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
19
20         jLabel1.setText("jLabel1");
21
22         jButton1.setText("jButton1");
23
24         jTextField1.setText("jTextField1");
25
```

```
26          javax.swing.GroupLayout layout = new
                javax.swing.GroupLayout(getContentPane());
27          getContentPane().setLayout(layout);
28          layout.setHorizontalGroup(
29              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
30              .addGroup(layout.createSequentialGroup()
31                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
32                      .addGroup(layout.createSequentialGroup()
33                          .addGap(77, 77, 77)
34                          .addComponent(jLabel1))
35                      .addGroup(layout.createSequentialGroup()
36                          .addGap(144, 144, 144)
37                          .addComponent(jButton1))
38                      .addGroup(layout.createSequentialGroup()
39                          .addGap(123, 123, 123)
40                          .addComponent(jTextField1,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
41                  .addContainerGap(183, Short.MAX_VALUE))
42          );
43          layout.setVerticalGroup(
44              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
45              .addGroup(layout.createSequentialGroup()
46                  .addGap(73, 73, 73)
47                  .addComponent(jLabel1)
48                  .addGap(26, 26, 26)
49                  .addComponent(jTextField1,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
50                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        50, Short.MAX_VALUE)
51                  .addComponent(jButton1)
52                  .addGap(94, 94, 94))
53          );
54
55          pack();
56      }// </editor-fold>
57
58      public static void main(String args[]) {
59          java.awt.EventQueue.invokeLater(new Runnable() {
60              public void run() {
61                  new VentanaUno().setVisible(true);
62              }
63          });
64      }
65
66      // Variables declaration - do not modify
67      private javax.swing.JButton jButton1;
68      private javax.swing.JLabel jLabel1;
69      private javax.swing.JTextField jTextField1;
70      // End of variables declaration
71
72  }
```

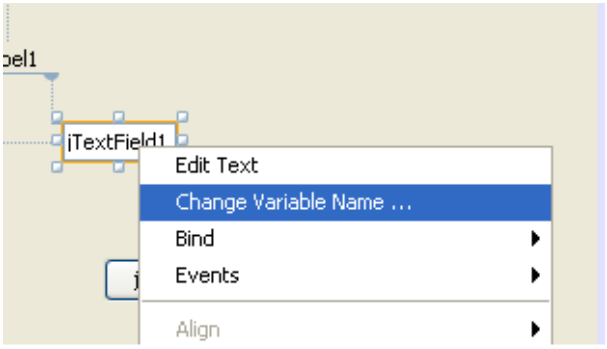
8.4. Componentes y eventos soportados por Java

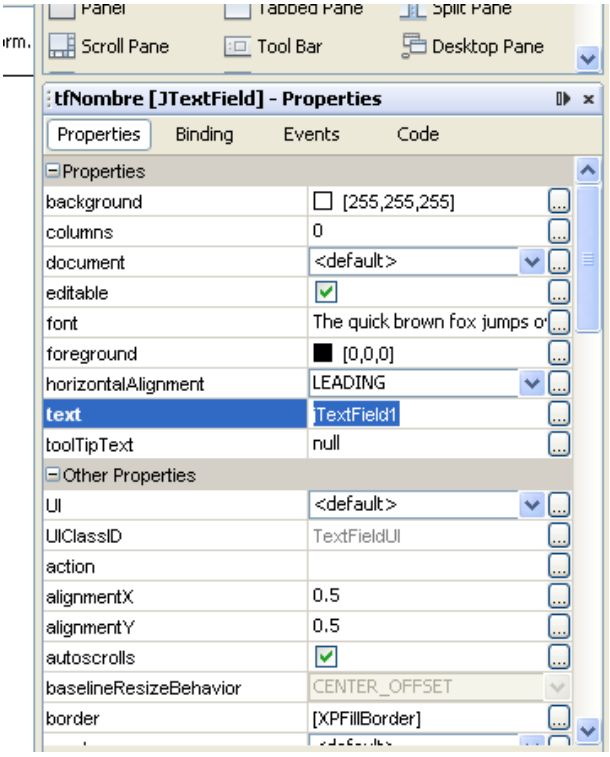


Clase	Descripcion
JButton	Botones con un texto
JChoice	Lista desplegable
JCheckBox	Casilla cuadrada para marcar
JScrollBar	Barra de desplazamiento
JTextField	Caja de texto
JList	Lista constantemente desplegada
JLabel	Etiqueta

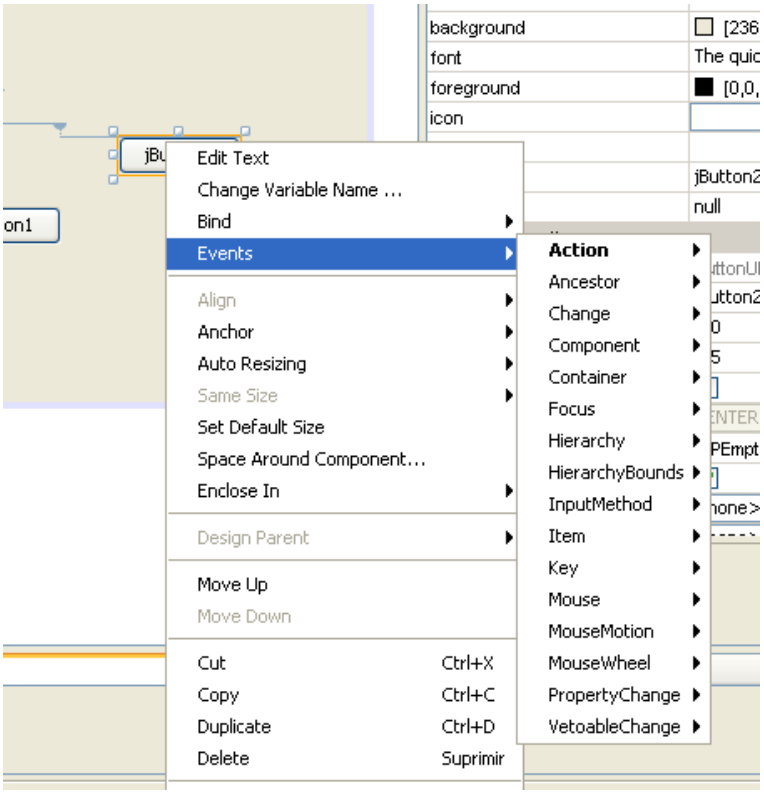
Al seleccionar en la paleta una clase y pinchar sobre nuestra ventana, estamos creando un objeto de ese tipo. Este objeto, como todos, dispone de un conjunto de atributos y métodos. Antes de acceder a ellos conviene poner un nombre más significativo. A partir de aquí, establecemos el valor de sus propiedades (atributos) y en ejecución podemos utilizar los métodos.

Dentro de las propiedades, tanto de las cajas de texto, como de los botones y etiquetas, destacar la propiedad **Text** que hace referencia al texto que muestran.


jTextField1
Edit Text
Change Variable Name ...
Bind
Events
Align


Scroll Pane
Tool Bar
Desktop Pane
tfNombre [JTextField] - Properties
Properties
Binding
Events
Code
Properties
background
[255,255,255]
columns
0
document
<default>
editable
font
The quick brown fox jumps o
foreground
[0,0,0]
horizontalAlignment
LEADING
text
TextField1
toolTipText
null
Other Properties
UI
<default>
UIClassID
TextFieldUI
action
alignmentX
0.5
alignmentY
0.5
autoscrolls
baselineResizeBehavior
CENTER_OFFSET
border
[XPFillBorder]

8.4.1. Eventos



Un evento es una acción que realiza el usuario sobre el interface.

ActionPerformed Se activa cuando el usuario pulsa INTRO.

mouseClicked Se activa cuando se hace click con ratón.

MouseMoved Se activa al mover por encima del objeto el ratón.

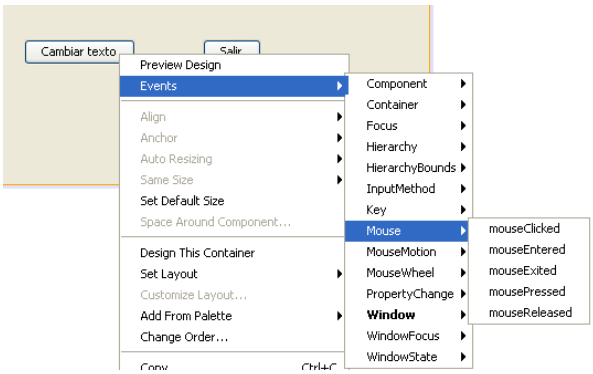
MouseDragged Se activa cuando se hace click (cualquier botón del ratón) sobre el objeto y se mueve el ratón por encima con el botón presionado.

FocusLost Se activa cuando pierde el foco o cursor.

FocusGained Se activa cuando gana el foco.

KeyPressed Se activa al presionar cualquier tecla sobre el objeto.

```
1
2       private void
            botonAceptarMouseClicked(java.awt.event.MouseEvent evt) {
3         ..............................
4     }
```




```
1  package Excepciones;
2
3  public class DniNoValidoException extends Exception{
4      public String getMessage(){
5          return "El dni tecleado no es valido";
6      }
7  }
```

```
1  package Excepciones;
2
3  public class NombreNoValidoException  extends Exception{
4      public String getMessage(){
5          return "El nombre tecleado no es valido";
6      }
7  }
```

```
1  package InterfazGrafico;
2
3  import Excepciones.*;
4  import ejemploventanas.Main;
5  import javax.swing.JOptionPane;
6
7  public class VentanaUno extends javax.swing.JFrame {
8
9      public VentanaUno() {
10         initComponents();
11     }
12
13
14     @SuppressWarnings("unchecked")
15     // <editor-fold defaultstate="collapsed" desc="Generated Code">
16     private void initComponents() {
17
18         jLabel1 = new javax.swing.JLabel();
19         jLabel2 = new javax.swing.JLabel();
20         jLabel3 = new javax.swing.JLabel();
21         tfDni = new javax.swing.JTextField();
22         tfNombre = new javax.swing.JTextField();
23         bAceptar = new javax.swing.JButton();
24         bCancelar = new javax.swing.JButton();
```

```
25
26          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
27
28          jLabel1.setText("Dni");
29
30          jLabel2.setText("Nombre");
31
32          jLabel3.setText("ALTA DE PERSONAS");
33
34          bAceptar.setText("Aceptar");
35          bAceptar.addActionListener(new
                java.awt.event.ActionListener() {
36              public void actionPerformed(java.awt.event.ActionEvent
                    evt) {
37                  bAceptarActionPerformed(evt);
38              }
39          });
40
41          bCancelar.setText("Cancelar");
42
43          javax.swing.GroupLayout layout = new
                javax.swing.GroupLayout(getContentPane());
44          getContentPane().setLayout(layout);
45          layout.setHorizontalGroup(
46              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
47              .addGroup(layout.createSequentialGroup()
48                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
49                      .addGroup(layout.createSequentialGroup()
50                          .addGap(57, 57, 57)
51                          .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
52                              .addComponent(jLabel1)
53                              .addComponent(jLabel2))
54                          .addGap(50, 50, 50)
55                          .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                                false)
56                              .addComponent(jLabel3)
57                              .addComponent(tfDni,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    173, Short.MAX_VALUE)
58                              .addComponent(tfNombre)))
59                      .addGroup(layout.createSequentialGroup()
60                          .addGap(83, 83, 83)
61                          .addComponent(bAceptar)
62                          .addGap(36, 36, 36)
63                          .addComponent(bCancelar)))
64                  .addContainerGap(83, Short.MAX_VALUE))
65          );
66          layout.setVerticalGroup(
67              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
68              .addGroup(layout.createSequentialGroup()
69                  .addGap(38, 38, 38)
70                  .addComponent(jLabel3)
71                  .addGap(54, 54, 54)
72                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
73                      .addComponent(jLabel1)
74                      .addComponent(tfDni,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
75                  .addGap(33, 33, 33)
76                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
77                      .addComponent(jLabel2)
78                      .addComponent(tfNombre,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
79                  .addGap(36, 36, 36)
80                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
81                      .addComponent(bAceptar)
82                      .addComponent(bCancelar))
83                  .addContainerGap(62, Short.MAX_VALUE))
84          );
85
86          pack();
87      }// </editor-fold>
88
89  private void bAceptarActionPerformed(java.awt.event.ActionEvent evt)
        {
90        try
91        {
92            validarDni();
93            validarNombre();
94            Main.altaPersona(tfDni.getText(),tfNombre.getText());
95        }
96        catch(DniNoValidoException e)
97        {
98            JOptionPane.showMessageDialog(this, e.getMessage());
99            // Colocar el cursor
100           tfDni.requestFocus();
101
102       }
103       catch(NombreNoValidoException e)
104       {
105           JOptionPane.showMessageDialog(this, e.getMessage());
106           tfNombre.requestFocus();
107       }
108       catch(Exception e)
109       {
110           JOptionPane.showMessageDialog(this, e.getMessage());
111           bCancelar.requestFocus();
112       }
113
114 }
115
116   public void validarDni() throws Exception{
117       if (tfDni.getText().isEmpty())
118           throw new DniNoValidoException();
119
120       if (Main.validarDni(tfDni.getText())== false)
121               throw new DniNoValidoException();
122
123   }
124
125   public boolean validarNombre(){
126       //Similar a validarDni
127       return true;
128   }
129
```

```
130
131         });
132     }
133     // Variables declaration - do not modify
134     private javax.swing.JButton bAceptar;
135     private javax.swing.JButton bCancelar;
136     private javax.swing.JLabel jLabel1;
137     private javax.swing.JLabel jLabel2;
138     private javax.swing.JLabel jLabel3;
139     private javax.swing.JTextField tfDni;
140     private javax.swing.JTextField tfNombre;
141     // End of variables declaration
142 }
```

```
1  package UML;
2
3  public class Persona {
4      private String Dni;
5      private String nombre;
6
7      public Persona(){}
8
9      public String getDni() {
10         return Dni;
11     }
12
13     public void setDni(String Dni) {
14         this.Dni = Dni;
15     }
16
17     public String getNombre() {
18         return nombre;
19     }
20
21     public void setNombre(String nombre) {
22         this.nombre = nombre;
23     }
24 }
```

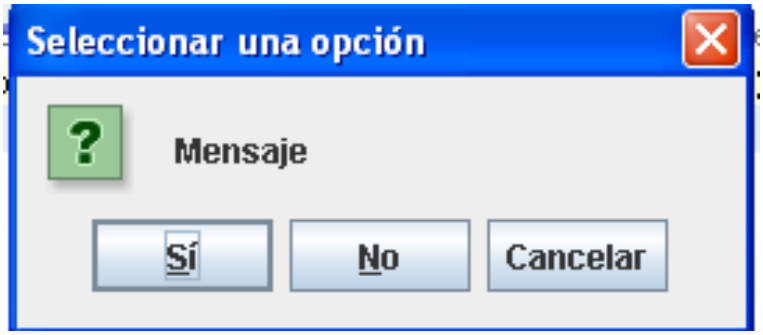
```
1  package ejemploventanas;
2
3  import UML.*;
4
5  public class Main {
6
7      private static Persona p;
8
9      public static void main(String[] args) {
10
11     }
12     public static boolean validarDni(String dni){
13         //Recorrer el arrayList y buscar o lo que sea preciso
14         return true;
15     }
16     public static void altaPersona(String dni, String nombre){
17         p = new Persona();
18         p.setDni(dni);
19         p.setNombre(nombre);
```

```
20          }
21  }
```

8.5. Cuadros de diálogo

Ya sabemos que los cuadros de diálogo se utilizan como ventanas que nos permiten llevar a cabo la entrada de un valor (JOptionPane.showInputDialog), para mostrar un mensaje al usuario (JOptionPane.showMessageDialog) o para permitir que nos indiquen la operación que se desea realizar (JOptionPane.showConfirmDialog).

```
1          int resultado =
               JOptionPane.showConfirmDialog(null,"Mensaje");
2
3         if ( resultado== 0)
4           botonMostrar.setText("Primer boton");
5        else
6          if ( resultado == 1)
7            botonMostrar.setText("Segundo boton");
8          else
9            botonMostrar.setText("Tercer boton");
```



Parámetros

Component Determina encima de que ventana se mostrará.

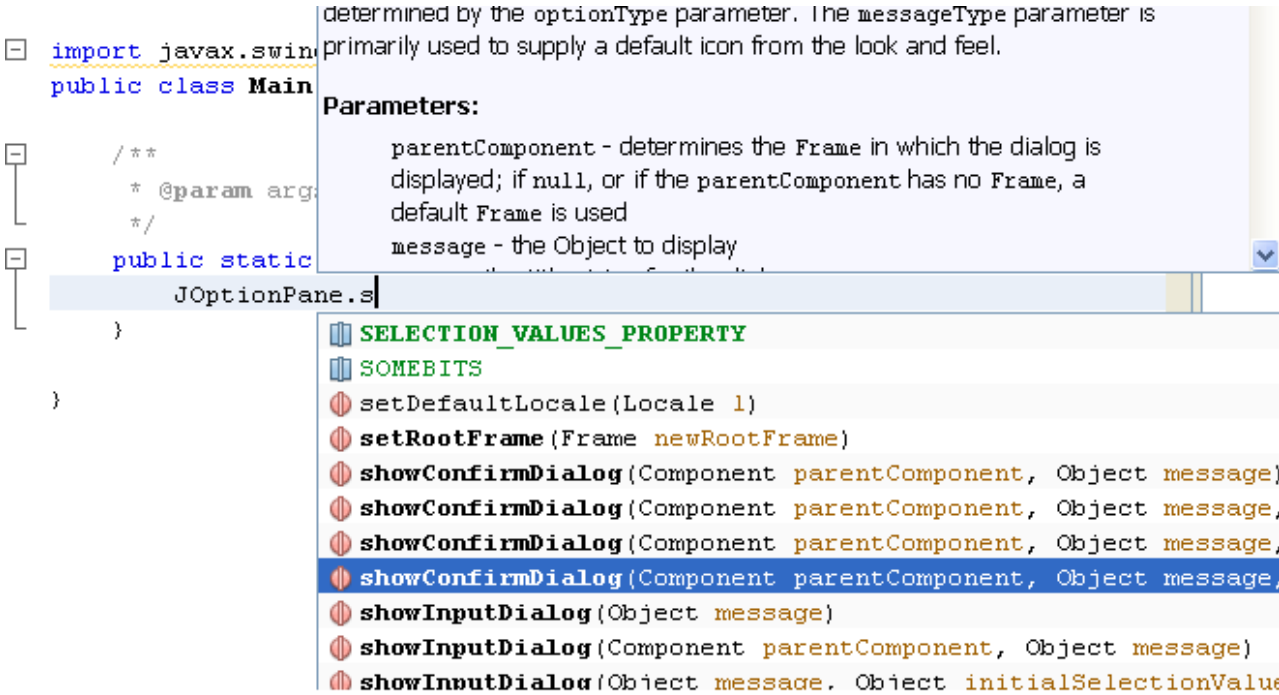
Message Mensaje de confirmación que deseamos visualizar.

title Título del cuadro.

OptionType(Botones que se van a mostrar) YES-NO-OPTION, YES-NO-CANCEL-OPTION, etc....

Tipo de mensaje ERROR-MESSAGE, INFORMATION-MESSAGE, WARNING-MESSAGE, PAIN-MESSAGE, QUESTION-MESSAGE, etc.. Determina el tipo de icono que se muestra.

Icono Icono que se muestra.



8.6. JDialog

Los dos tipos de ventanas principales que tenemos en java son JFrame y JDialog.

Los JDialog son ideales para ventanas secundarias porque admiten una ventana padre. Si la VentanaA es padre del JDialogB, entonces el JDialogB siempre estará por delante de VentanaA, nunca quedará por detrás. Lo ideal es que hagamos nuestras ventanas secundarias como JDialog cuyo padre sea el JFrame principal. De esta forma los JDialog siempre serán visibles por encima del JFrame y no se irán detrás ni quedarán ocultos por el JFrame.

Un JDialog puede ser modal, pasando un true en el constructor en el sitio adecuado o haciéndolo modal con el método setModal(). Si hacemos un JDialog modal, todas las demás ventanas se deshabilitarán hasta que el usuario de nuestro programa cierre el JDialog. Esto está bien para pedir un dato al usuario y evitar que toque otras cosas hasta que haya introducido el dato.

```
1  private void botonActionPerformed(java.awt.event.ActionEvent evt) {
2          new Dialogo( this, false).setVisible(true);
```

```
3       }
```

```
1   public class Dialogo extends javax.swing.JDialog {
2
3       public Dialogo(java.awt.Frame parent, boolean modal) {
4           super(parent, modal);
5           initComponents();
6       }
7
8       @SuppressWarnings("unchecked")
9       // <editor-fold defaultstate="collapsed" desc="Generated Code">
10      private void initComponents() {
11
12          jLabel1 = new javax.swing.JLabel();
13          jScrollPane1 = new javax.swing.JScrollPane();
14          jTextArea1 = new javax.swing.JTextArea();
15          jScrollPane2 = new javax.swing.JScrollPane();
16          jList1 = new javax.swing.JList();
17          jButton1 = new javax.swing.JButton();
18
19          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
20
21          jLabel1.setText("jLabel1");
22
23          jTextArea1.setColumns(20);
24          jTextArea1.setRows(5);
25          jScrollPane1.setViewportView(jTextArea1);
26
27          jList1.setModel(new javax.swing.AbstractListModel() {
28              String[] strings = { "Item 1", "Item 2", "Item 3", "Item
                    4", "Item 5" };
29              public int getSize() { return strings.length; }
30              public Object getElementAt(int i) { return strings[i]; }
31          });
32          jScrollPane2.setViewportView(jList1);
33
34          jButton1.setText("jButton1");
35
36          javax.swing.GroupLayout layout = new
                javax.swing.GroupLayout(getContentPane());
37          getContentPane().setLayout(layout);
38          layout.setHorizontalGroup(
39              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
40              .addGroup(layout.createSequentialGroup()
41                  .addGap(92, 92, 92)
42                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
43                      .addComponent(jScrollPane1,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
44                      .addComponent(jLabel1))
45                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        62, Short.MAX_VALUE)
46                  .addComponent(jScrollPane2,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
47                  .addGap(45, 45, 45))
48              .addGroup(layout.createSequentialGroup()
```

```
49                  .addGap(135, 135, 135)
50                  .addComponent(jButton1)
51                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
52          );
53          layout.setVerticalGroup(
54              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
55              .addGroup(layout.createSequentialGroup()
56                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
57                      .addGroup(layout.createSequentialGroup()
58                          .addGap(66, 66, 66)
59                          .addComponent(jLabel1)
60                          .addGap(18, 18, 18)
61                          .addComponent(jScrollPane1,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                35,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
62                      .addGroup(layout.createSequentialGroup()
63                          .addGap(44, 44, 44)
64                          .addComponent(jScrollPane2,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
65                  .addGap(9, 9, 9)
66                  .addComponent(jButton1)
67                  .addContainerGap(94, Short.MAX_VALUE))
68          );
69
70          pack();
71      }// </editor-fold>
72
73
74
75          /* Create and display the dialog */
76          java.awt.EventQueue.invokeLater(new Runnable() {
77              public void run() {
78                  dialogo dialog = new dialogo(new
                        javax.swing.JFrame(), true);
79                  dialog.addWindowListener(new
                        java.awt.event.WindowAdapter() {
80                      @Override
81                      public void
                            windowClosing(java.awt.event.WindowEvent e) {
82                          System.exit(0);
83                      }
84                  });
85                  dialog.setVisible(true);
86              }
87          });
88      }
89
90      // Variables declaration - do not modify
91      private javax.swing.JButton jButton1;
92      private javax.swing.JLabel jLabel1;
93      private javax.swing.JList jList1;
94      private javax.swing.JScrollPane jScrollPane1;
95      private javax.swing.JScrollPane jScrollPane2;
96      private javax.swing.JTextArea jTextArea1;
97      // End of variables declaration
```

```
98  }
```

8.7. Clase Window

Los objetos de la clase Window son ventanas de máximo nivel, pero sin bordes y sin barra de menús. Son más interesantes las clases que derivan de ella: JFrame y JDialog. Los métodos más útiles, que son heredados por estas dos clases son los siguientes:

void windowOpened(WindowEvent we); // antes de mostrarla por primera vez

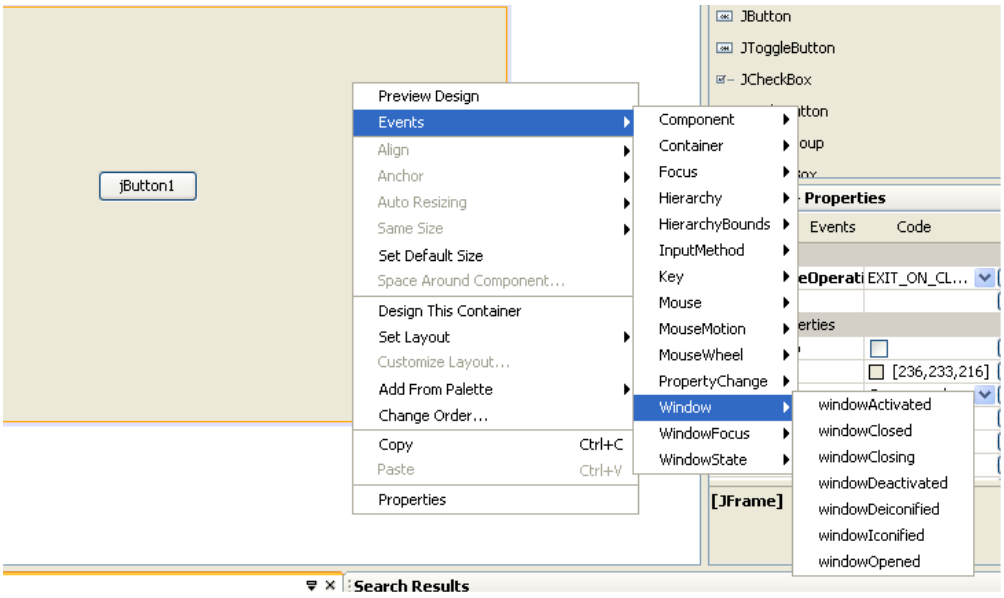
void windowClosing(WindowEvent we); // al recibir una solicitud de cierre

void windowClosed(WindowEvent we); // después de cerrar la ventana

void windowIconified(WindowEvent we);

void windowDeiconified(WindowEvent we);

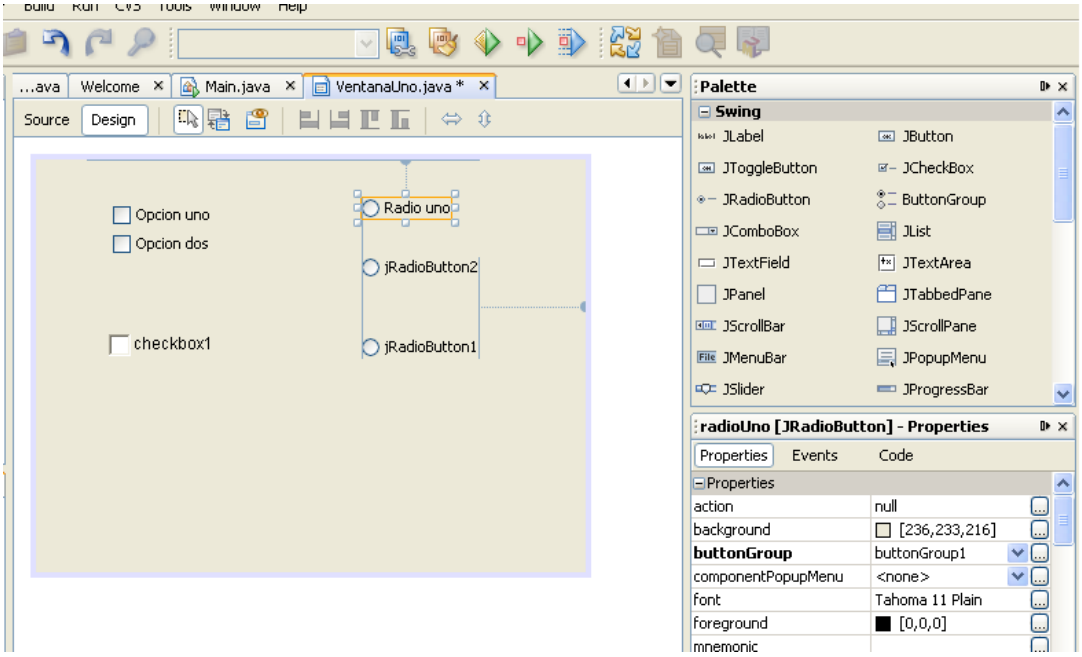
void windowActivated(WindowEvent we);



8.8. Checkbox, RadioButton y ButtonGroup

Los objetos de las clases Checkbox y RadioButton son botones de opción o de selección con dos posibles valores: true o false. Si tenemos varios checkbox, existe la posibilidad de seleccionar varios. Si tenemos varios radioButton en un **grupo** (ButtonGroup), sólo se puede seleccionar uno de ellos. Al cambiar la selección de un Checkbox se produce un ItemEvent.

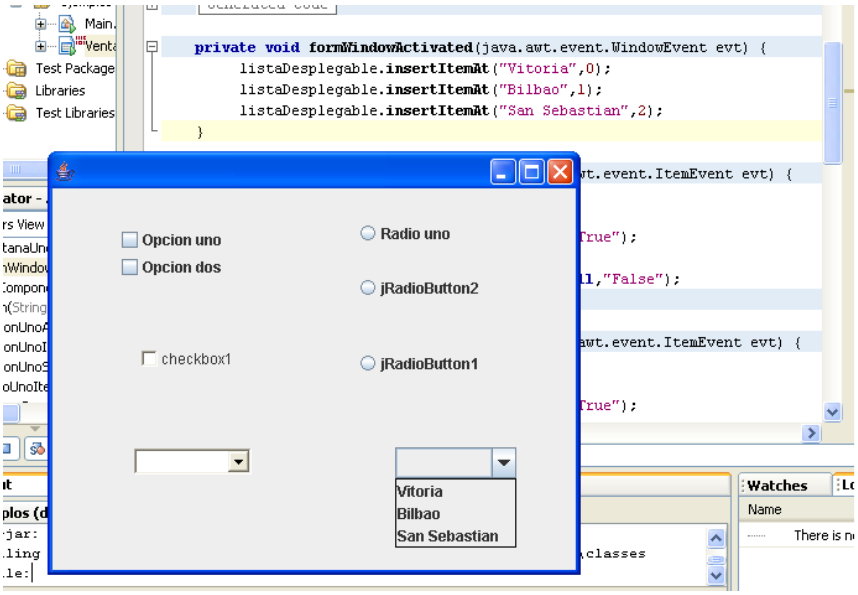
```
1  private void opcionUnoItemStateChanged(java.awt.event.ItemEvent
       evt) {
2          if (opcionUno.isSelected())
3              JOptionPane.showConfirmDialog(null,"True");
4          else
5                  JOptionPane.showConfirmDialog(null,"False");
6      }
```



8.9. Clase Choice o comboBox

La clase Choice permite elegir un elemento o item de una lista desplegable. Los objetos Choice ocupan menos espacio en pantalla que los Checkbox. Al elegir un item se genera un ItemEvent. Un index permite determinar un elemento de la lista (se empieza a contar desde 0). (setSelectedIndex(), setSelectedItem())

Si en diseño conocemos los datos que debemos mostrar, estos se establecen en la propiedad **model**. Si no es así, es decir, si queremos introducir los elementos en ejecución, debemos utilizar métodos **insert**.



8.10. JPanel

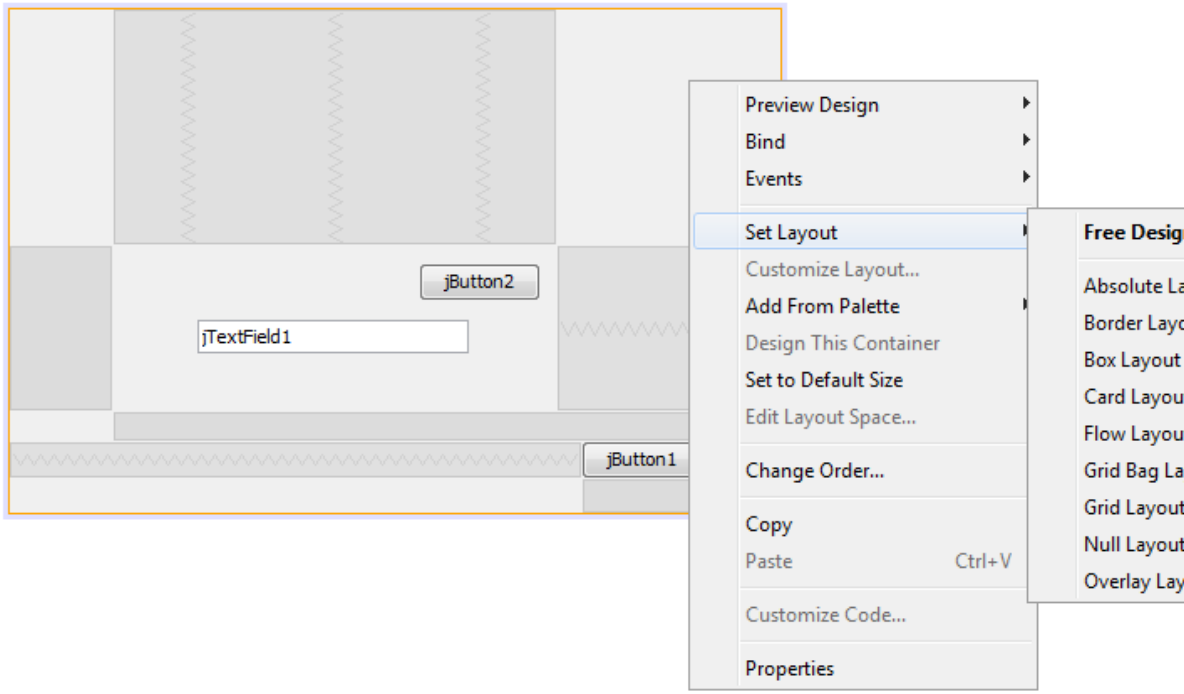
El JFrame es la ventana (en windows, la que tiene la X a la derecha y arriba) mientras que el JPanel es una especie de **marco contenedor** donde puedes meter cualquier objeto gráfico válido.

Un JFrame puede contener uno o más paneles; de hecho trae uno por defecto al que se puede acceder mediante **getContentPane()**. Un jPanel sólo puede contener otros paneles (además de los otros objetos como botones o etiquetas), el panel no puede contener frames.

```
1  public class panelUno extends javax.swing.JPanel {
2
3      public panelUno() {
4          initComponents();
5      }
6
7
8      @SuppressWarnings("unchecked")
9      // <editor-fold defaultstate="collapsed" desc="Generated Code">
10     private void initComponents() {
11
12         c = new javax.swing.JButton();
13         d = new javax.swing.JButton();
14
15         c.setText("jButton1");
16
17         d.setText("jButton2");
18
19         javax.swing.GroupLayout layout = new
               javax.swing.GroupLayout(this);
20         this.setLayout(layout);
21         layout.setHorizontalGroup(
22             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
23             .addGroup(layout.createSequentialGroup()
24                 .addGap(40, 40, 40)
25                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
26                     .addComponent(d)
27                     .addComponent(c))
28                 .addContainerGap(287, Short.MAX_VALUE))
29         );
30         layout.setVerticalGroup(
31             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
32             .addGroup(layout.createSequentialGroup()
33                 .addGap(55, 55, 55)
34                 .addComponent(c)
35                 .addGap(18, 18, 18)
36                 .addComponent(d)
37                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                       Short.MAX_VALUE))
38         );
39     }// </editor-fold>
40
41
42     // Variables declaration - do not modify
43     private javax.swing.JButton c;
```

```
44      private javax.swing.JButton d;
45      // End of variables declaration
46  }
```

8.11. Layout



En java, cuando creamos ventanas, la clase que decide cómo se reparten los botones y demás controles se llama Layout. Esta clase es la que decide en qué posición van los botones y demás componentes, si van alineados, en forma de matriz, cuáles se hacen grandes al agrandar la ventana, etc. Otra cosa importante que decide el Layout es qué tamaño es el ideal para la ventana en función de los componentes que lleva dentro.

Las ventanas vienen con un Layout por defecto. En java hay varios layouts disponbiles y podemos cambiar el de defecto por el que queramos.

8.11.1. El Layout null

Uno de los layouts más utilizados por la gente que empieza, por ser el más sencillo, es NO usar layout. Somos nosotros desde código los que decimos cada botón en qué posición va y qué tamaño ocupa.

Esto, aunque sencillo, no es recomendable. Si estiramos la ventana los componentes seguirán en su sitio, es decir, no se estirarán con la

ventana. Si cambiamos de sistema operativo, resolución de pantalla o fuente de letra, tenemos casi asegurado que no se vean bien las cosas: etiquetas cortadas, letras que no caben, etc.

El tiempo que ahorramos no aprendiendo cómo funcionan los layouts, lo perderemos echando cuentas con los pixels, para conseguir las cosas donde queremos, sólo para un tipo de letra y un tamaño fijo.

8.11.2. FlowLayout

El FlowLayout es bastante sencillo de usar. Nos coloca los componente en fila. Hace que todos quepan (si el tamaño de la ventana lo permite). Es adecuado para barras de herramientas, filas de botones, etc.

8.11.3. BoxLayout

Es como un FlowLayout, pero mucho más completo. Permite colocar los elementos en horizontal o vertical.

Un tutorial completo en `http://java.sun.com/docs/books/tutorial/uiswing/layout/box.html`

8.11.4. GridLayout

Este pone los componentes en forma de matriz (cuadrícula), estirándolos para que tengan todos el mismo tamaño. El GridLayout es adecuado para hacer tableros, calculadoras en que todos los botones son iguales, etc.

8.11.5. BorderLayout

El BorderLayout divide la ventana en 5 partes: centro, arriba, abajo, derecha e izquierda.

Hará que los componentes que pongamos arriba y abajo ocupen el alto que necesiten, pero los estirará horizontalmente hasta ocupar toda la ventana.

Los componentes de derecha e izquierda ocuparán el ancho que necesiten, pero se les estirará en vertical hasta ocupar toda la ventana.

El componente central se estirará en ambos sentidos hasta ocupar toda la ventana.

El BorderLayout es adecuado para ventanas en las que hay un componente central importante (una tabla, una lista, etc) y tiene menús

o barras de herramientas situados arriba, abajo, a la derecha o a la izquierda.

Este suele ser el layout por defecto para los JFrame y JDialog.

8.11.6. GridBagLayout

El GridBagLayout es de los layouts más versátiles y complejos de usar. Es como el GridLayout, pone los componentes en forma de matriz (cuadrícula), pero permite que las celdas y los componentes en ellas tengan tamaños variados.

Es posible hacer que un componente ocupe varias celdas.

Un componente puede estirarse o no con su celda.

Si no se estira, puede quedar en el centro de la celda o pegarse a sus bordes o esquinas.

Las columnas pueden ensancharse o no al estirar la ventana y la proporición podemos decidirla. Lo mismo con la filas.

8.11.7. CardLayout

El CardLayout hace que los componente recibidos ocupen el máximo espacio posible, superponiendo unos a otros. Sólo es visible uno de los componentes, los otros quedan detrás. Tiene métodos para indicar cual de los componentes es el que debe quedar encima y verse.

El CardLayout es el que utiliza el JTabbedPane (el de las pestañas) de forma que en función de la pestaña que pinchemos, se ve uno u otro.

8.11.8. SpringLayout

Se añaden los componentes y para cada uno de ellos tenemos que decir qué distancia en pixel queremos que tenga cada uno de sus bordes respecto al borde de otro componente.

Con este layout, cuando estiramos el panel, siempre ceden aquellos componentes más 'flexibles'. Entre una etiqueta y una caja de texto, la caja de texto es la que cambia su tamaño.

Los dos tipos de ventanas principales que tenemos en java son JFrame y JDialog.

Los JDialog son ideales para ventanas secundarias porque admiten una ventana padre. Si la VentanaA es padre del JDialogB, entonces el

JDialogB siempre estará por delante de VentanaA, nunca quedará por detrás. Lo ideal es que hagamos nuestras ventanas secundarias como JDialog cuyo padre sea el JFrame principal. De esta forma los JDialog siempre serán visibles por encima del JFrame y no se irán detrás ni quedarán ocultos por el JFrame.

Un JDialog puede ser modal, pasándole un true en el constructor en el sitio adecuado o haciéndolo modal con el método setModal(). Si hacemos un JDialog modal, todas las demás ventanas se deshabilitarán hasta que el usuario de nuestro programa cierre el JDialog. Esto está bien para pedir un dato al usuario y evitar que toque otras cosas hasta que haya introducido el dato.

```
1  private void botonActionPerformed(java.awt.event.ActionEvent evt) {
2          new dialogo( this, false).setVisible(true);
3      }
```

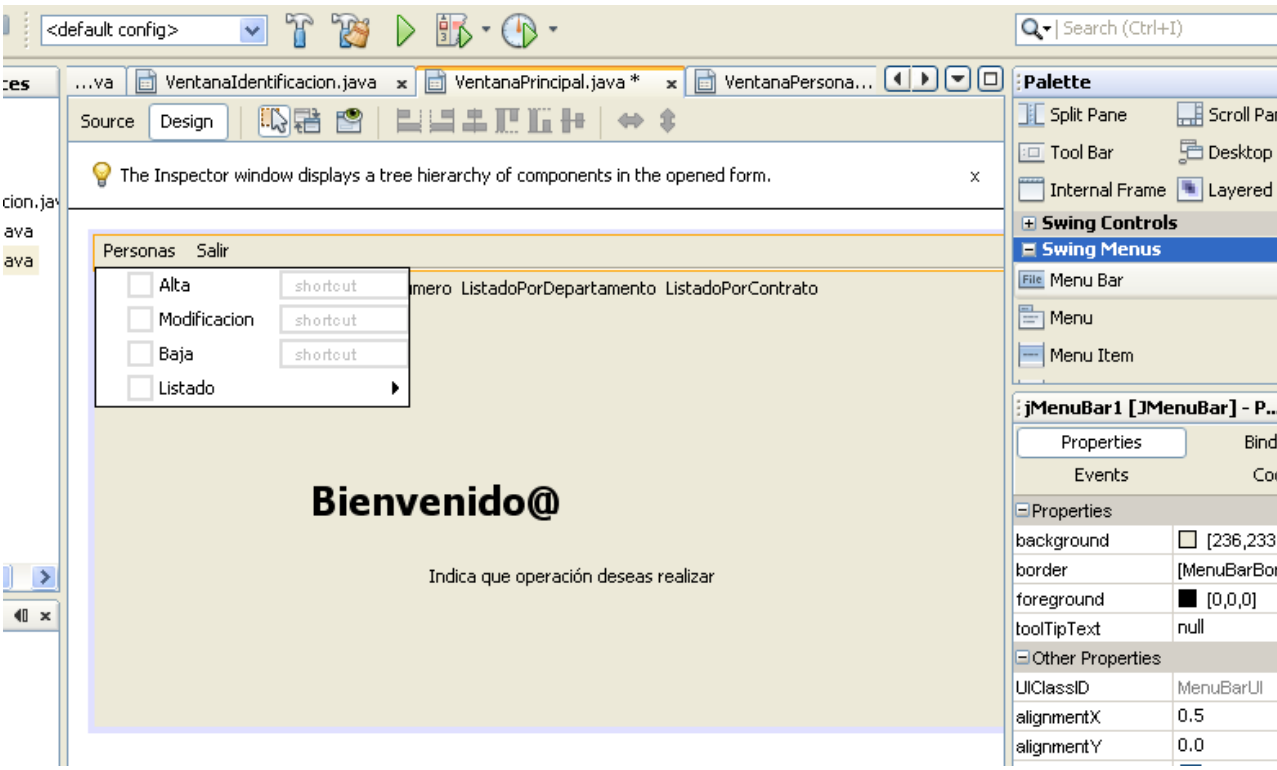
```
1  public class dialogo extends javax.swing.JDialog {
2
3      public dialogo(java.awt.Frame parent, boolean modal) {
4          super(parent, modal);
5          initComponents();
6      }
7
8      @SuppressWarnings("unchecked")
9      // <editor-fold defaultstate="collapsed" desc="Generated Code">
10     private void initComponents() {
11
12         jLabel1 = new javax.swing.JLabel();
13         jScrollPane1 = new javax.swing.JScrollPane();
14         jTextArea1 = new javax.swing.JTextArea();
15         jScrollPane2 = new javax.swing.JScrollPane();
16         jList1 = new javax.swing.JList();
17         jButton1 = new javax.swing.JButton();
18
19         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
20
21         jLabel1.setText("jLabel1");
22
23         jTextArea1.setColumns(20);
24         jTextArea1.setRows(5);
25         jScrollPane1.setViewportView(jTextArea1);
26
27         jList1.setModel(new javax.swing.AbstractListModel() {
28             String[] strings = { "Item 1", "Item 2", "Item 3", "Item
                   4", "Item 5" };
29             public int getSize() { return strings.length; }
30             public Object getElementAt(int i) { return strings[i]; }
31         });
32         jScrollPane2.setViewportView(jList1);
33
34         jButton1.setText("jButton1");
35
36         javax.swing.GroupLayout layout = new
               javax.swing.GroupLayout(getContentPane());
```

```
37          getContentPane().setLayout(layout);
38          layout.setHorizontalGroup(
39              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
40              .addGroup(layout.createSequentialGroup()
41                  .addGap(92, 92, 92)
42                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
43                      .addComponent(jScrollPane1,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
44                      .addComponent(jLabel1))
45                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        62, Short.MAX_VALUE)
46                  .addComponent(jScrollPane2,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
47                  .addGap(45, 45, 45))
48              .addGroup(layout.createSequentialGroup()
49                  .addGap(135, 135, 135)
50                  .addComponent(jButton1)
51                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
52          );
53          layout.setVerticalGroup(
54              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
55              .addGroup(layout.createSequentialGroup()
56                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
57                      .addGroup(layout.createSequentialGroup()
58                          .addGap(66, 66, 66)
59                          .addComponent(jLabel1)
60                          .addGap(18, 18, 18)
61                          .addComponent(jScrollPane1,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                35,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
62                      .addGroup(layout.createSequentialGroup()
63                          .addGap(44, 44, 44)
64                          .addComponent(jScrollPane2,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
65                  .addGap(9, 9, 9)
66                  .addComponent(jButton1)
67                  .addContainerGap(94, Short.MAX_VALUE))
68          );
69
70          pack();
71      }// </editor-fold>
72
73
74
75          /* Create and display the dialog */
76          java.awt.EventQueue.invokeLater(new Runnable() {
77              public void run() {
78                  dialogo dialog = new dialogo(new
                        javax.swing.JFrame(), true);
79                  dialog.addWindowListener(new
                        java.awt.event.WindowAdapter() {
```

```
80                      @Override
81                      public void
                            windowClosing(java.awt.event.WindowEvent e) {
82                          System.exit(0);
83                      }
84                  });
85                  dialog.setVisible(true);
86              }
87          });
88      }
89
90      // Variables declaration - do not modify
91      private javax.swing.JButton jButton1;
92      private javax.swing.JLabel jLabel1;
93      private javax.swing.JList jList1;
94      private javax.swing.JScrollPane jScrollPane1;
95      private javax.swing.JScrollPane jScrollPane2;
96      private javax.swing.JTextArea jTextArea1;
97      // End of variables declaration
98  }
```

8.12. Menús y barras de herramientas

Para crear un **menú** debemos utilizar la clase JMenuBar (barra de menú) que se encuentra dentro de la paleta en el apartado **Menús Swing**. Esta clase nos permitirá crear una barra de menú en la que a continuación debemos añadir menus (JMenu) y/o elementos de menú (JMenuItem) normales, de tipo casilla de verifiación o de tipo botón de opción. Cada menú también dispondrá sus elementos.



Una vez creado el menú, de cara a la programación podremos utilizar los eventos ya vistos para los botones (MouseClicked, ActionPerformed, etc..).

Para crear una **barra de herramientas**, debemos utilizar la clase JToolBar situada en Contenedores swing. Una vez creada la barra de herramientas, en ella añadiremos los botones que consideremos oportunos.

Una vez añadidos los botones, a la hora de programar utilizaremos los mismos eventos que cuando los botones no están en una barra.

Es muy habitual colocar en una barra de herramientas los botones que

permitan rápidamente acceder a las tareas más habituales. Estas tareas, también habitualmente, están colocadas en el menú que contiene todas las opciones. A la hora de codificar las operaciones a realizar, se deben codificar sólo en un lugar, por ejemplo, en el menú. A continuación, en el click del botón situado en la barra de herramientas, se codificará la orden que provoque la ejecución del método asociado a la opción del menú.

```
1  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
2  {
3                OpcionUno.doClick();
4  }
```