Construcción de Interficies Gráficas en Java

Projecte de Programació (PROP)

AWT – *Abstract Window Toolkit*

- Parte de Java que gestiona las interficies gráficas

- Resto de gestores de interficies se basan o relacionan con AWT

- Conceptos básicos:

  - Componentes: objetos y contenedores de la interficie

  - *Layout Managers*: regulan la distribución de componentes en contenedores

  - Eventos: transmitidos por el sistema operativo al AWT cuando el usuario realiza una acción

AWT: Componentes

- Componentes primarios (*widgets*):
  - Label: texto no editable
  - TextArea, TextField: texto editable
  - Button: botones
  - Choice, Checkbox, List: selección de una o varias opciones
  - Scrollbar: barra de desplazamiento
  - Canvas: zona de imágenes

AWT: Componentes

- Componentes contenedores:

  - Contenedores principales:

    No pueden estar dentro de otro contenedor

    - Frame: el contenedor más importante

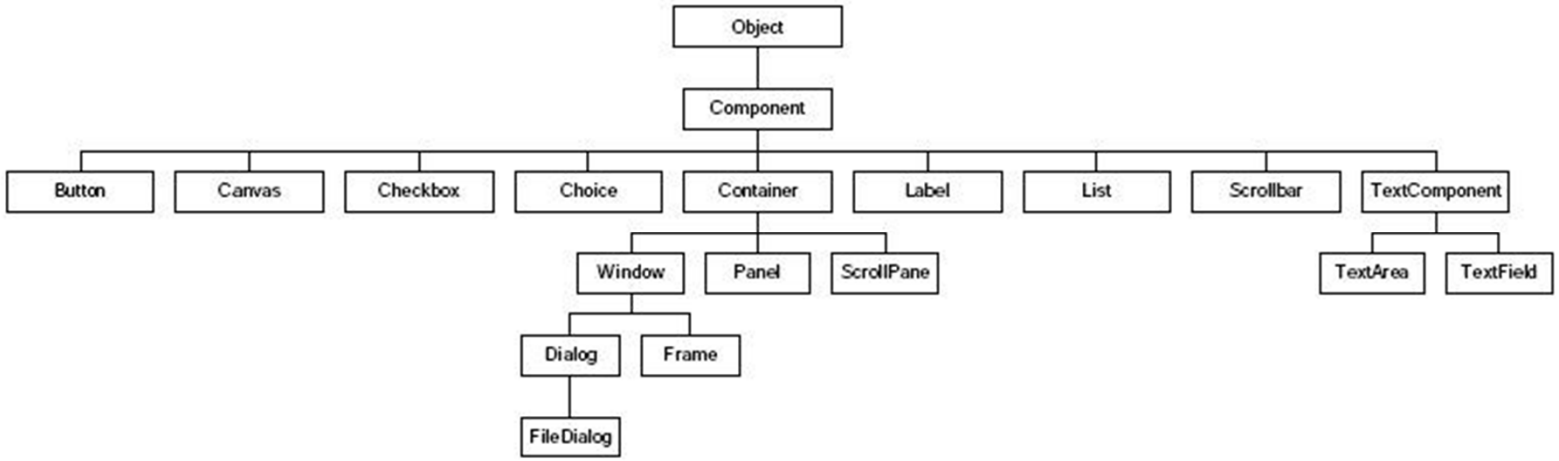
    - Dialog: depende de un frame – dialog modal

  - Contenedores secundarios:

    Han de estar dentro de otro contenedor

    Pueden contener a otros contenedores secundarios

    - Panel (Applet es subclase de él), ScrollPane



AWT: Componentes

- Reglas de funcionamiento:
  - Un componente sólo puede estar en un contenedor
  - Interficie funcional: hay que tener un contenedor principal, y hacerlo visible:

    miContainer.setVisible(true)
  - Añadir un componente a un contenedor:

    miContainer.add(miComponent)
  - Eliminar un componente de un contenedor:

    miContainer.remove(miComponent)

- Una interficie funcional será un árbol n-ario:
  - Raíz: un Frame
  - Nodos intermedios: contenedores secundarios
  - Hojas: componentes primarios

AWT: Layout Managers

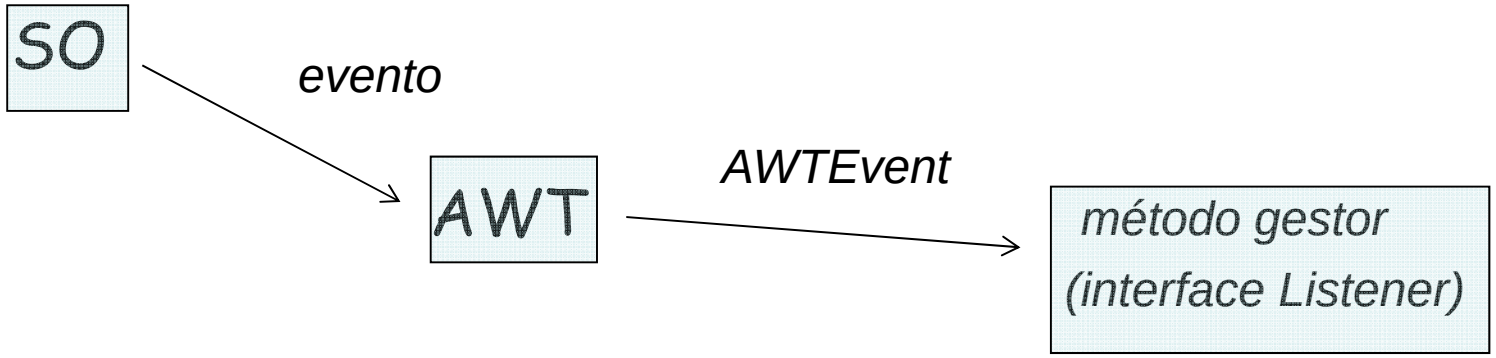
- Controlan cómo se distribuyen los componentes en los contenedores (principales y secundarios).

- Se pueden definir personalizados o usar uno de los 5 predefinidos:
  - *FlowLayout*: los componentes se van colocando de izquierda a derecha y arriba abajo
  - BorderLayout: los componentes se colocan en 5 zonas (North, South, East, West, Center)
  - *GridLayout*: los componentes se colocan en una matriz de celdas del mismo tamaño
  - *GridBagLayout*: similar al anterior pero con componentes que pueden ocupar más de una celda
  - *CardLayout*: el mismo espacio puede ser usado por elementos diferentes en momentos diferentes

- *Layouts* por defecto:
  - Contenedores principales: *BorderLayout*
  - Contenedores secundarios: *FlowLayout*

- Para cambiar el *layout*: *miContainer.setLayout(new GridLayout());*

AWT: Eventos



- Asociar a cada componente *C* un objeto (o varios) *L* (*interface Listener*).
- *L* define un método *M* que recibe como parámetro un objeto subclase de *AWTEvent*.
- *M* se ejecuta cuando se produce el evento.

AWT: Eventos

```
public class MiClase() {
   // En alguna parte de la clase (al crearla, por ejemplo)
   public MiClase() {
        ...
        // Crea el listener del evento
        ListenerBoton miListenerBoton = new ListenerBoton();
        // Asocia el listener al boton
        miBoton.addActionListener(miListenerBoton);
        ...
   }
   ...
   // La clase ListenerBoton es privada
   private class ListenerBoton implements ActionListener {
        public void actionPerformed (ActionEvent event) {
                System.exit(0); // CODIGO ASOCIADO AL EVENTO
        }
   }
}
```

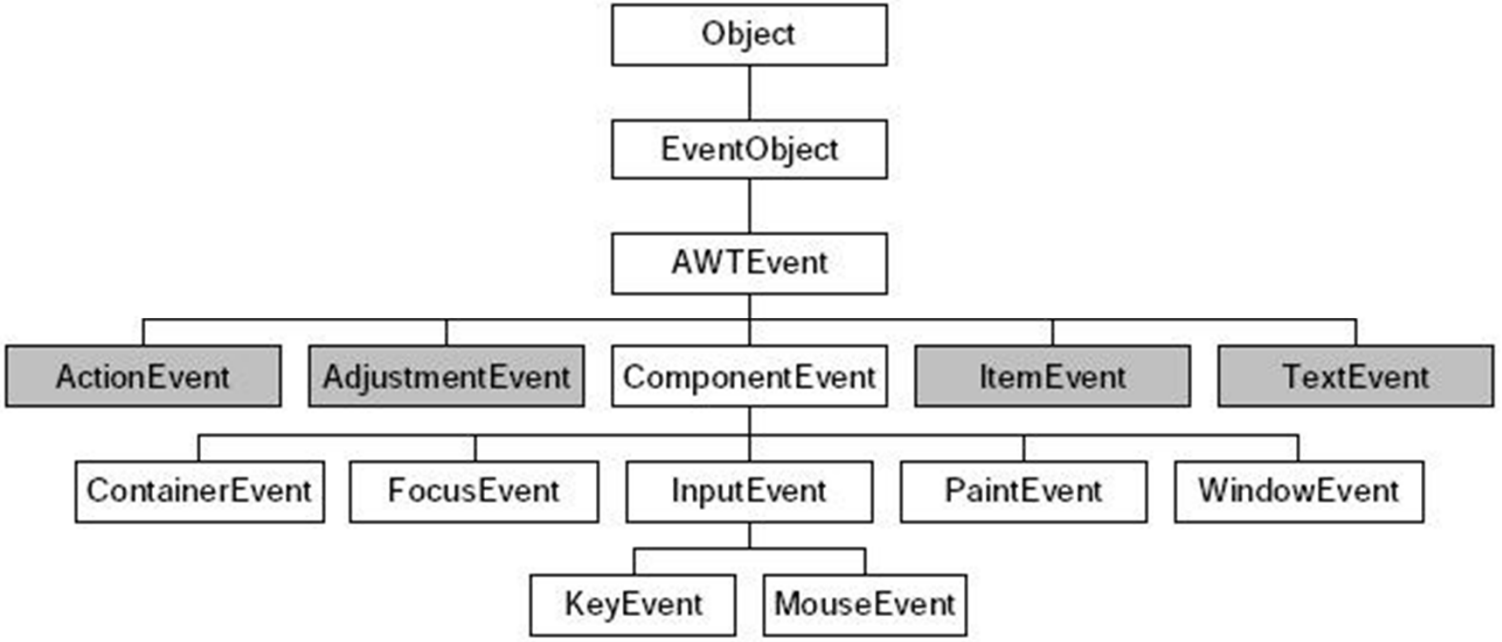
AWT: Eventos

Alternativa: usar clases anónimas

```
public class MiClase() {
   // En alguna parte de la clase (al crearla, por ejemplo)
   public MiClase() {
        ...
        // Crea y asocial el listener al botón
        miBoton.addActionListener
          (new ActionListener() {
             public void actionPerformed (ActionEvent event) {
                System.exit(0); // CODIGO ASOCIADO AL EVENTO
             }
          });
        ...
   }
   ...
}
```

AWT: Eventos

- De bajo nivel: operaciones elementales con ratón, teclado, contenedores y ventanas
- De alto nivel: operaciones con un significado en el contexto gráfico
  - Clickar en un botón
  - Cambiar las barras de desplazamiento
  - Elegir un valor entre varios
  - Cambiar un texto



AWT: Componentes vs Eventos

Component	Eventos generados	Significado
Button	ActionEvent	Clicar en el botón
Checkbox	ItemEvent	Seleccionar o deseleccionar un item
CheckboxMenuItem	ItemEvent	Seleccionar o deseleccionar un item
Choice	ItemEvent	Seleccionar o deseleccionar un item
Component	ComponentEvent	Mover, cambiar tamaño, mostrar u ocultar un componente
	FocusEvent	Obtener o perder el focus
	KeyEvent	Pulsar o soltar una tecla
	MouseEvent	Pulsar o soltar un botón del ratón; entrar o salir de un componente; mover o arrastrar el ratón (tener en cuenta que este evento tiene dos Listener)
Container	ContainerEvent	Añadir o eliminar un componente de un container
List	ActionEvent	Hacer doble click sobre un item de la lista
	ItemEvent	Seleccionar o deseleccionar un item de la lista
MunuItem	ActionEvent	Seleccionar un item de un menú
Scrollbar	AdjustementEvent	Cambiar el valor de la scrollbar
TextComponent	TextEvent	Cambiar el texto
TextField	ActionEvent	Terminar de editar un texto pulsando Intro
Window	WindowEvent	Acciones sobre una ventana: abrir, cerrar, iconizar, restablecer e iniciar el cierre

AWT: Eventos vs Listeners

- Evento *XxxEvent* <-> Listener *XxxListener*

Evento	Interface Listener	Métodos de Listener
ActionEvent	ActionListener	actionPerformed()
AdjustementEvent	AdjustementListener	adjustementValueChanged()
ComponentEvent	ComponentListener	componentHidden(), componentMoved(), componentResized(), componentShown()
ContainerEvent	ContainerListener	componentAdded(), componentRemoved()
FocusEvent	FocusListener	focusGained(), focusLost()
ItemEvent	ItemListener	itemStateChanged()
KeyEvent	KeyListener	keyPressed(), keyReleased(), keyTyped()
MouseEvent	MouseListener	mouseClicked(), mouseEntered(), mouseExited(), mousePressed(), mouseReleased()
	MouseMotionListener	mouseDragged(), mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowActivated(), windowDeactivated(), windowClosed(), windowClosing(), windowIconified(), windowDeiconified(), windowOpened()

- *Micomponent.addXxxListener (objetoXxxListener)*
- *Micomponent.removeXxxListener (objetoXxxListener)*

- Clases *Adapter*: contienen la implementación vacía de cada operación de un *Listener.* Heredando de ellas y redefiniendo sólo los métodos necesarios ahorramos codificar.

AWT: Menús

- Se pueden asociar a un contenedor principal en forma de "barra de menú". 3 clases:
  - MenuItem: opción que permite la ejecución de una acción
  - Menu: agrupa varios MenuItem u otros Menus
  - MenuBar: barra en el contenedor principal

- Casos particulares:
  - CheckboxMenuItem: items que se pueden activar o no
  - PopupMenu: aparecen al hacer click con el botón derecho del ratón

AWT: Menús

- Para tener una barra de menús funcional:
  - Crear los elementos de cada tipo

```
MenuItem miMenuItem = new MenuItem("Menu Option");
Menu miMenu = new Menu("Menu Name");
MenuBar miMenuBar = new MenuBar();
```

  - Asociarlos entre sí (operación *add*)

```
miMenu.add(miMenuItem);
miMenuBar.add(miMenu);
```

  - Asociar la MenuBar al contenedor principal

```
miFrame.setMenuBar(miMenuBar);
```

  - Asociar un ActionListener a cada MenuItem

```
miMenuItem.addActionListener
    (new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            … // Codigo asociado al evento
        }
    });
```

JAVA: Threads

- Flujos secuenciales de ejecución dentro de un proceso
- Tipos:
  - Thread inicial: inicia la ejecución de la aplicación
  - Event-dispatching thread: donde se procesan las tareas gráficas del AWT una vez activados sus componentes
  - Worker o background threads: para tareas costosas en segundo plano
- Para sincronizar threads:
  - **`wait([time])`**: bloquea el objeto al que se aplica el método, deteniendo la ejecución del thread en el que está
  - **`notify()`**: desbloquea el objeto, se puede activar uno de los threads que le esperaban
  - **`notifyAll()`**: desbloquea el objeto, se activan todos los threads que esperaban

AWT: Event-dispatching Thread

- En una aplicación basada en interficie gráfica, sólo el lanzamiento de la vistaPrincipal se hace desde el thread inicial

```
public static void main (String args[]) {
  vistaPrincipal = new VistaPrincipal();
  ... // Inicializaciones
  vistaPrincipal.hacerVisible();
  System.out.printl("Seguramente no tiene sentido poner nada aquí…");
}
```

- A partir de ese momento toman el control los Listeners asociados a los componentes gráficos: event-dispatching thread
- Si se quiere que alguna tarea costosa se ejecute sin dejar la parte gráfica bloqueada, se puede crear uno o más working threads

AWT: Event-dispatching Thread

- Si se quiere ejecutar todo desde el event-dispatching thread, se puede crear un objeto de la interficie Runnable y "encolarlo" en él:

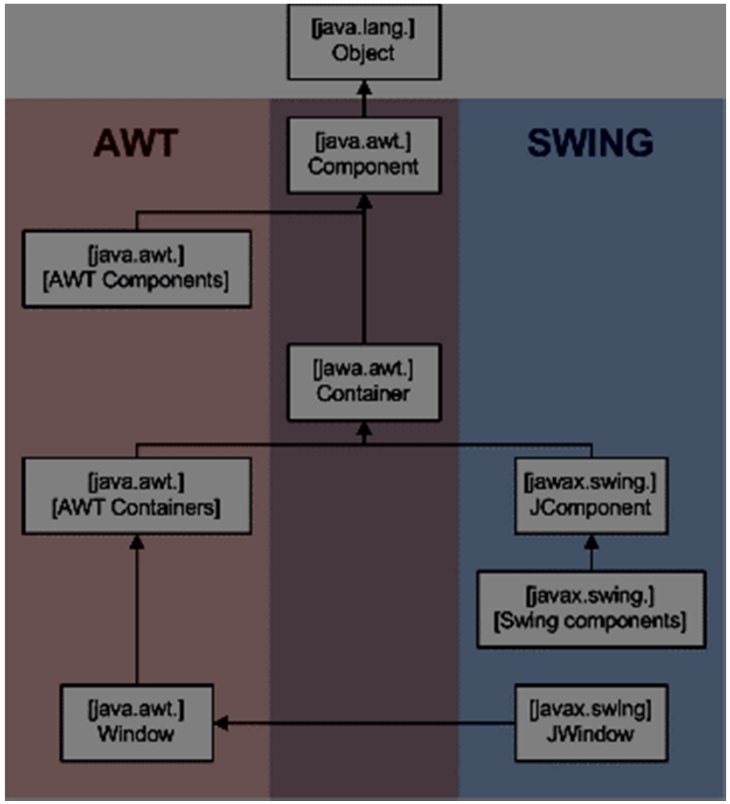
```
public static void main (String args[]) {
   java.awt.EventQueue.invokeLater (
        new Runnable() {
              public void run() {
                    vistaPrincipal = new VistaPrincipal();
                    ... // Inicializaciones
                    vistaPrincipal.hacerVisible();
              }
        }
   );
}
```

Swing

- Extensión más habitual de AWT, incorporada en los entornos de desarrollo
- Ofrece mayor gama de componentes:
  - JTabbedPane: contenedor con pestañas
  - JTree: para representar árboles
  - JComboBox: listas seleccionables flexibles
  - JTable: para representar tablas
  - JProgressBar: para mostrar el progreso de la ejecución
  - …
- Amplia los métodos y eventos disponibles
- Amplia funcionalidades:
  - Asociar iconos a componentes
  - Tooltips: sugerencias o comentarios
  - Toolbars: barras con botones
  - …

Swing vs AWT

- Componentes primarios y contenedores secundarios heredan de *JComponent:*
  - Componentes primarios: JButton, JMenuItem, JComboBox, JFileChooser, JLabel, JList, JMenuBar, JScrollBar, JTextArea, JTextField, …
  - Contenedores secundarios: Jpanel, JScrollPane, JOptionPane, …
- Contenedores principales (*JFrame* y *JDialog*) heredan de sus equivalentes en AWT



Swing vs AWT

- El funcionamiento general es igual (con una *J* delante)

  Árbol n-ario en cuya raíz hay un JFrame, en los nodos intermedios contenedores secundarios y en las hojas componentes primarios

- Se entrelazan ciertas clases. Por ejemplo, para construir ciertos diálogos es mejor no usar JDialog, sino:
  - JOptionPane para establecer diálogos rápidos ("Yes/No", "Yes/No/Cancel", …)
  - JFileChooser para seleccionar un fichero
  - JColorChooser para escoger un color

- Los métodos de AWT son thread-safe, los de Swing en general no -> el event-dispatching thread debe ser el único thread válido para modificar los estados de las componentes gráficas una vez que están activas (es lo que ocurre por defecto)

Swing vs AWT

- En Swing no hay que asociar un evento al cerrar un JFrame, basta con asignar la operación por defecto:

miJFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

Otras opciones disponibles: DO NOTHING ON CLOSE, HIDE ON CLOSE, DISPOSE ON CLOSE

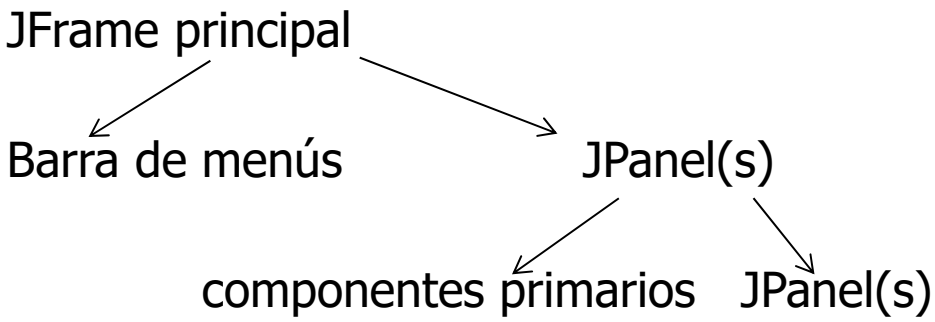
- Al crear un JFrame se crean automáticamente los contenedores:
  - rootPane: contiene al resto de contenedores
  - layeredPane: contiene la barra de menús y el contentPane
  - contentPane: contiene los componentes visibles

```
JPanel contentPane = (JPanel)miJFrame.getContentPane();
contentPane.setLayout(new FlowLayout());
contentPane.add(miBoton);
```

  - glassPane: para capturar eventos de áreas que contienen uno o más componentes

Swing: Implementación Típica

- Elementos habituales:

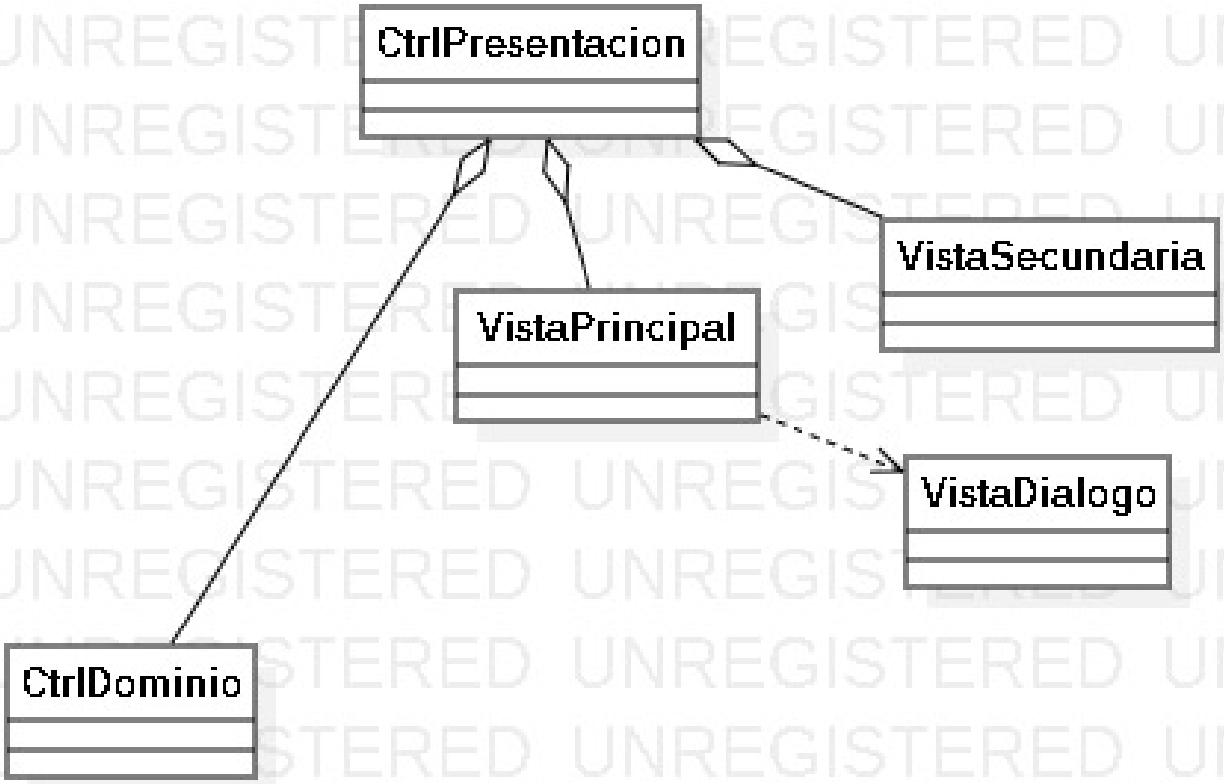


- Visibilidad si hay varios JPanels:
  - -Todos visibles (algunos quizá desactivados) -> añadir todos al contentPane
  - -Sólo un subconjunto (o uno) visible cada vez -> ir modificando el contentPane con el JPanel correspondiente:
    `miJFrame.setContentPane(miJPanel)`

- Layouts:
  - -A nivel de JPanel, para colocar las componentes en el JPanel
  - -A nivel de JFrame, para colocar los JPanel en el JFrame

Swing: Implementación Típica

Un `main()` en una clase MainEjemplo lanza el programa. Este `main()` podría haber estado en el propio CtlPresentacion



Swing: Implementación Típica (I)

1.Para iniciar la ejecución, el **`main()`** crea un objeto de la clase CtrlPresentacion e inicia la presentación:

```
public class MainEjemplo {
     public static void main (String[] args) {
        javax.swing.SwingUtilities.invokeLater (
                new Runnable() {
                   public void run() {
                     CtrlPresentacion ctrlPresentacion =
                                          new CtrlPresentacion();
                     ctrlPresentacion.inicializarPresentacion();
                   }
                }
        );
     }
}
```

Swing: Implementación Típica (II)

2. El CtrlPresentacion:
   - Crea una instancia del controlador de dominio y la vista principal
   - Inicializa el controlador de dominio y hace visible la vista
   - Contiene los métodos para sincronizar las vistas y llamar al controlador del dominio
   - En este ejemplo pasaremos el CtrlPresentacion como parámetro a la vista para que desde los eventos que ocurren en la vista se traten sus operaciones asociadas:

```
public class CtrlPresentacion {
  private CtrlDominio ctrlDominio;
  private VistaPrincipal vistaPrincipal;

  public CtrlPresentacion() {
        ctrlDominio = new CtrlDominio();
        vistaPrincipal = new VistaPrincipal(this);
  }

  public void inicializarPresentacion()
        ctrlDominio.inicializarCtrlDominio();
        vistaPrincipal.hacerVisible();
  }
  ...
}
```

Swing: Implementación Típica (III)

3. La VistaPrincipal contiene un atributo privado JFrame (alternativamente podría heredar de él) y atributos privados para todos sus componentes:

```
public class VistaPrincipal {
  private CtrlPresentacion iCtrlPresentacion;
  private JFrame frameVista = new JFrame("Vista Principal");
  private JPanel panelContenidos = new JPanel();
  …
  private JButton buttonLlamadaDominio = new JButton("Llamada Dominio");
  private JLabel labelPanelInformacion1 = new JLabel("Panel Informacion 1");
  private JComboBox comboboxInformacion1 = new JComboBox();
  private JTextArea textareaInformacion1 = new JTextArea(15,25);
  private JTextField textfieldInformacion2 = new JTextField();
  private JSlider sliderInformacion2 = new JSlider();
  …
  private JMenuBar menubarVista = new JMenuBar();
  private JMenu menuFile = new JMenu("File");
  private JMenuItem menuitemQuit = new JMenuItem("Quit");
   …
  public VistaPrincipal (CtrlPresentacion pCtrlPresentacion) {
     iCtrlPresentacion = pCtrlPresentacion;
     inicializarComponentes();
  }
…
```

Swing: Implementación Típica (IV)

4. El método *inicializarComponentes()* :
   - Define las características del JFrame y los componentes
   - Define y asocia los layout managers
   - Añade los componentes a sus contenedores correspondientes
   - Define y asocia los Listeners a los componentes

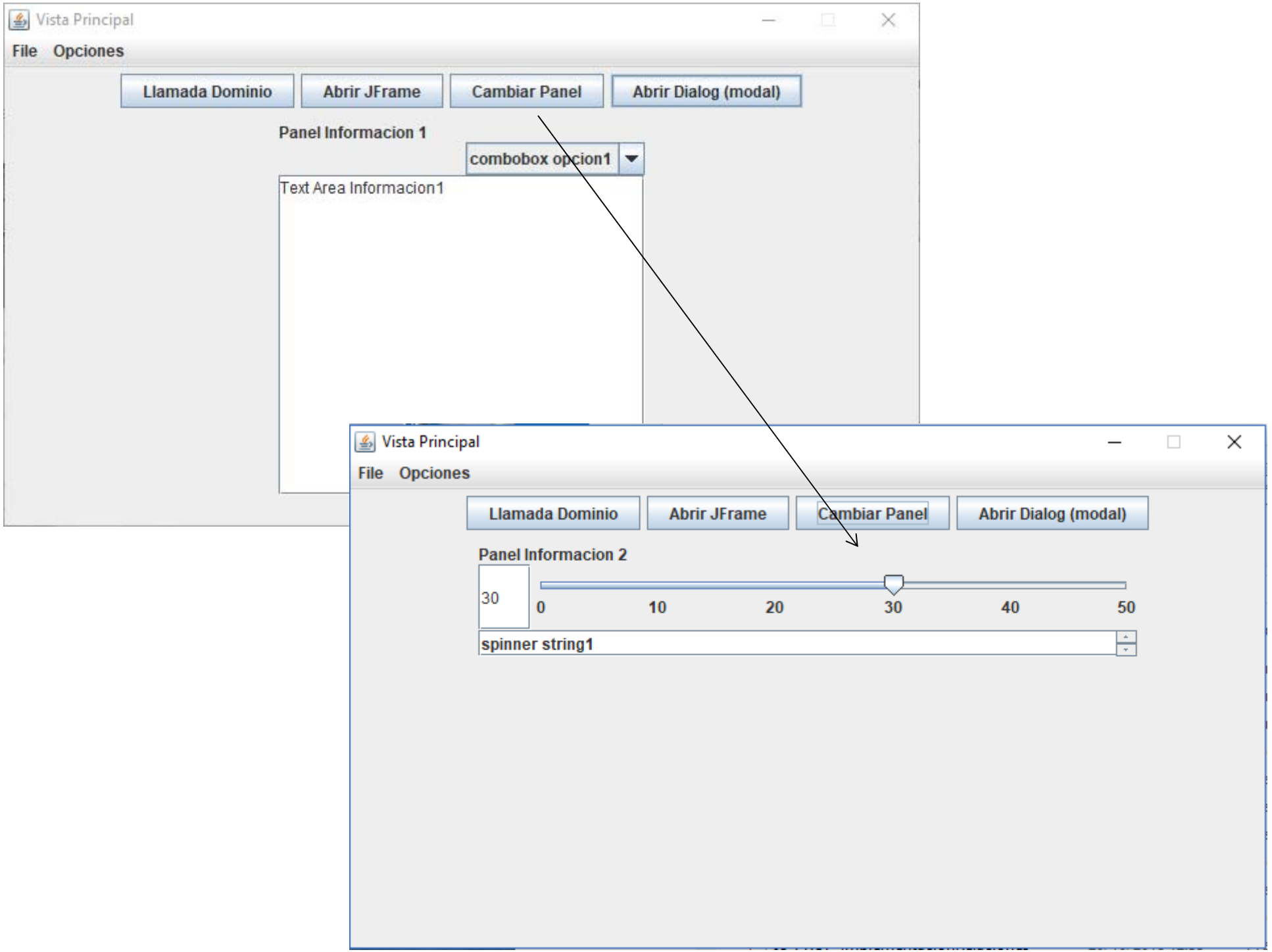
```
buttonLlamadaDominio.addActionListener
       (new ActionListener() {
         public void actionPerformed (ActionEvent event) {
           String texto = ((JButton) event.getSource()).getText();
           System.out.println("Has clickado el boton con texto: " +
    texto);
           actionPerformed_buttonLlamadaDominio(event);
         }
       });
```

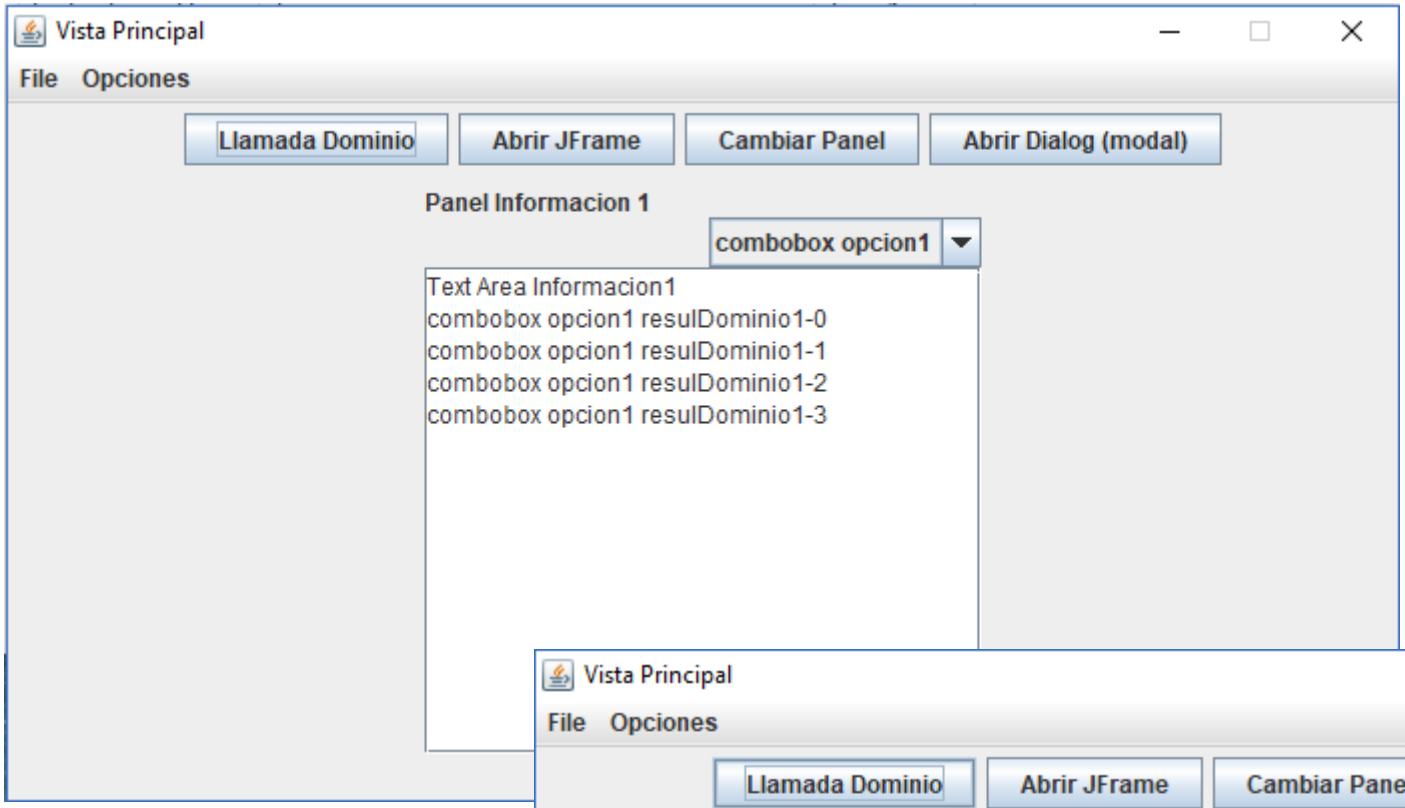
   - Asigna el primer panel visible a *contentPane* del Jframe (si hay varios, si hubiera uno bastaría con asociar los componentes al *contentPane* directamente)
   - Posiciona el frame en la pantalla (vía *setLocation/setLocationRelativeTo*)

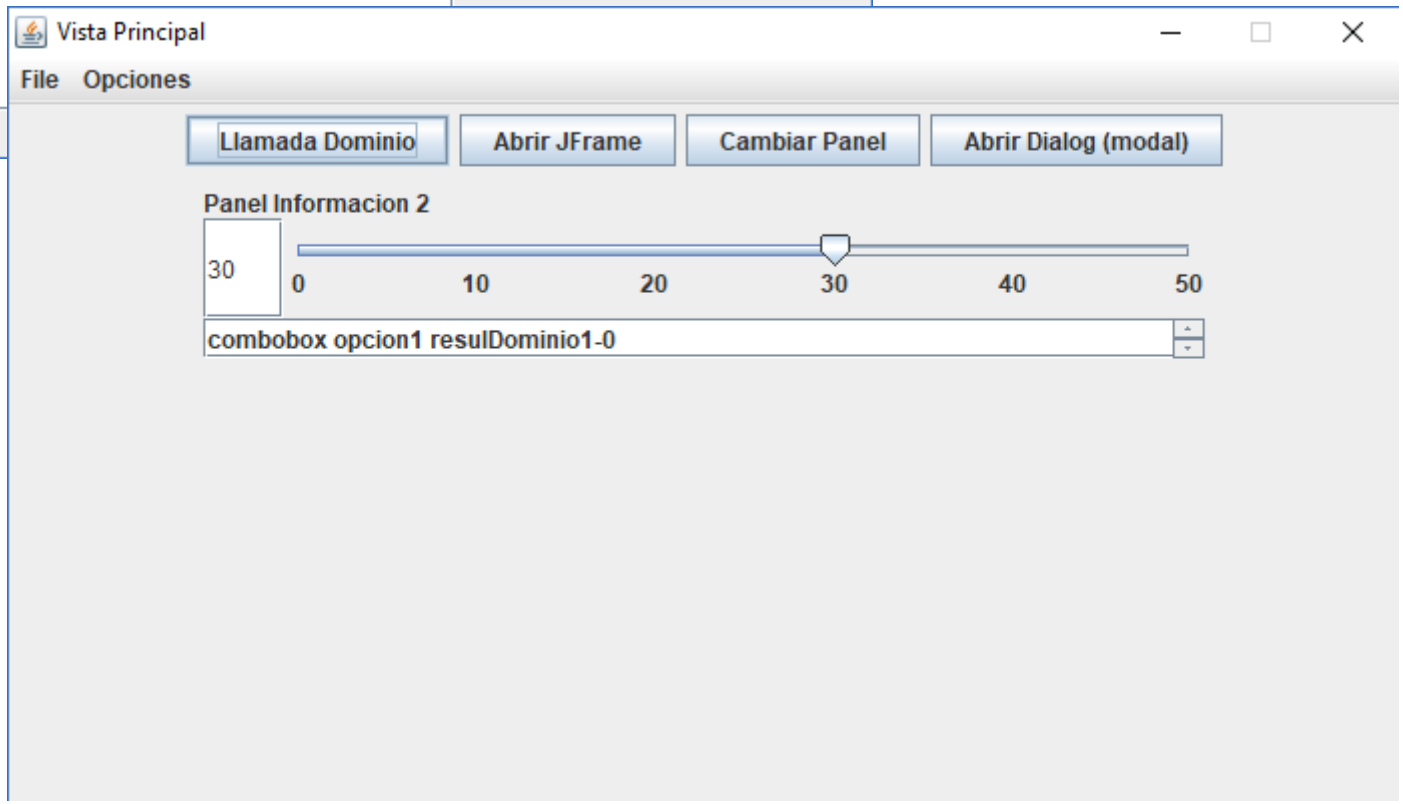
Swing: Implementación Típica (V)

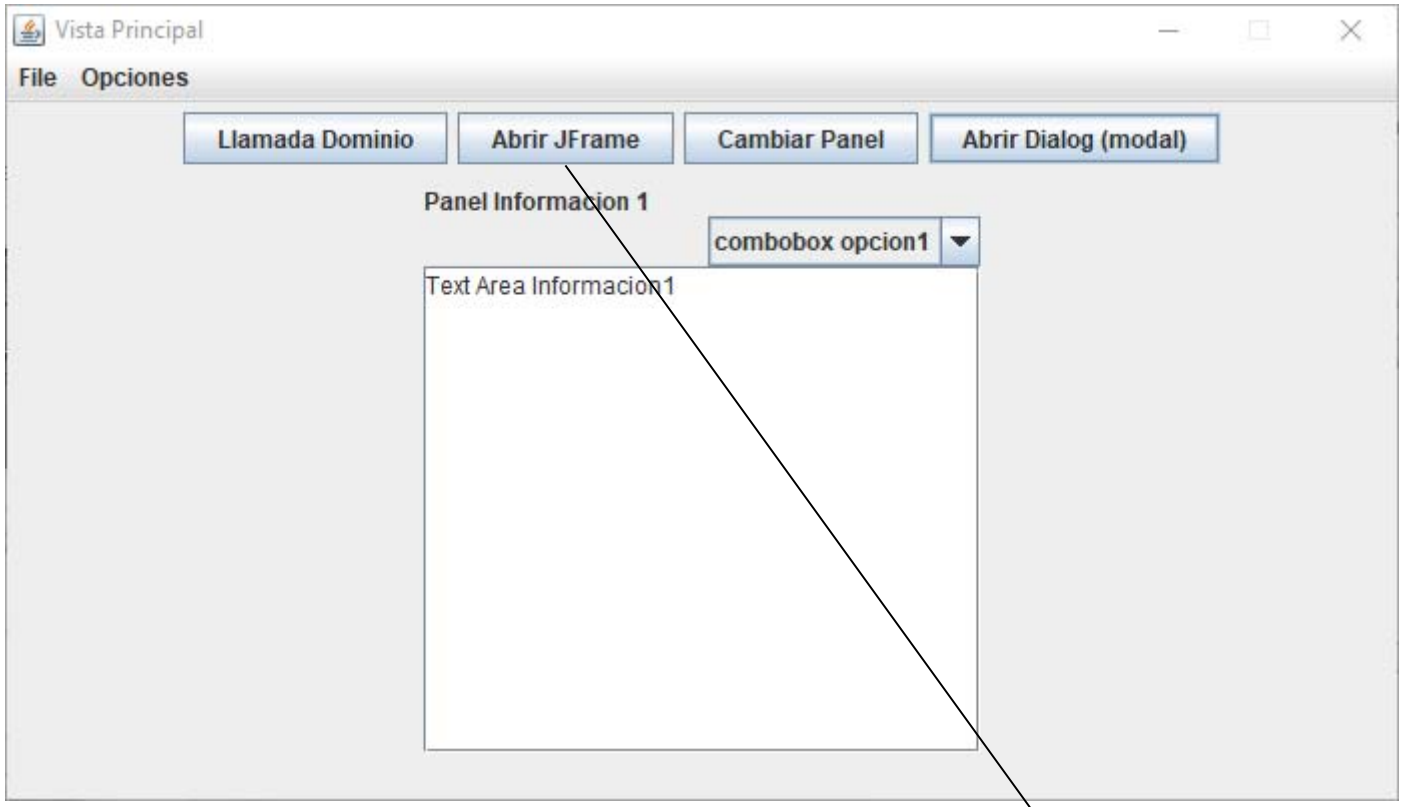
5. Los métodos que se llaman en cada Listener se definen también en la vista, pero actúan llamando a las operaciones del controlador de presentación:

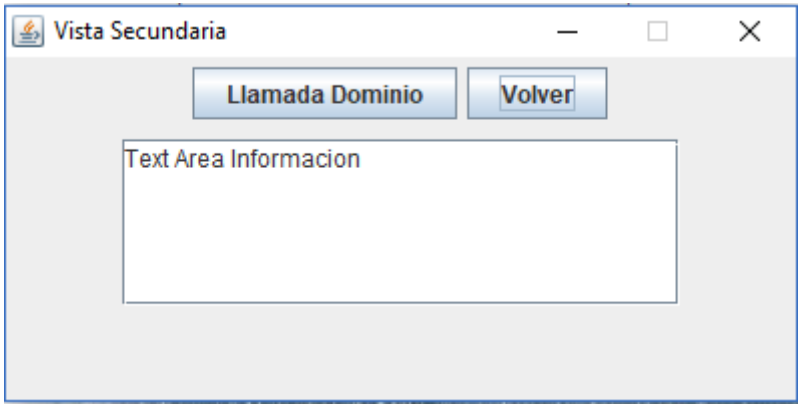
```
public void actionPerformed_buttonLlamadaDominio (ActionEvent event) {
    String comboboxSelectedItem =
      (String) comboboxInformacion1.getModel().getSelectedItem();
    ArrayList<String> resulDominio =
      iCtrlPresentacion.llamadaDominio1(comboboxSelectedItem);
    for (int i = 0; i < resulDominio.size(); i++)
      System.out.println("Obtenido de dominio: " + resulDominio.get(i));
    // Informa el contenido de algunos componentes (es un ejemplo)
    for (int i = 0; i < resulDominio.size(); i++)
      textareaInformacion1.append("\n" + resulDominio.get(i));
    SpinnerListModel spinnerModel =
      (SpinnerListModel) spinnerInformacion2.getModel();
    spinnerModel.setList(resulDominio);
  }
```

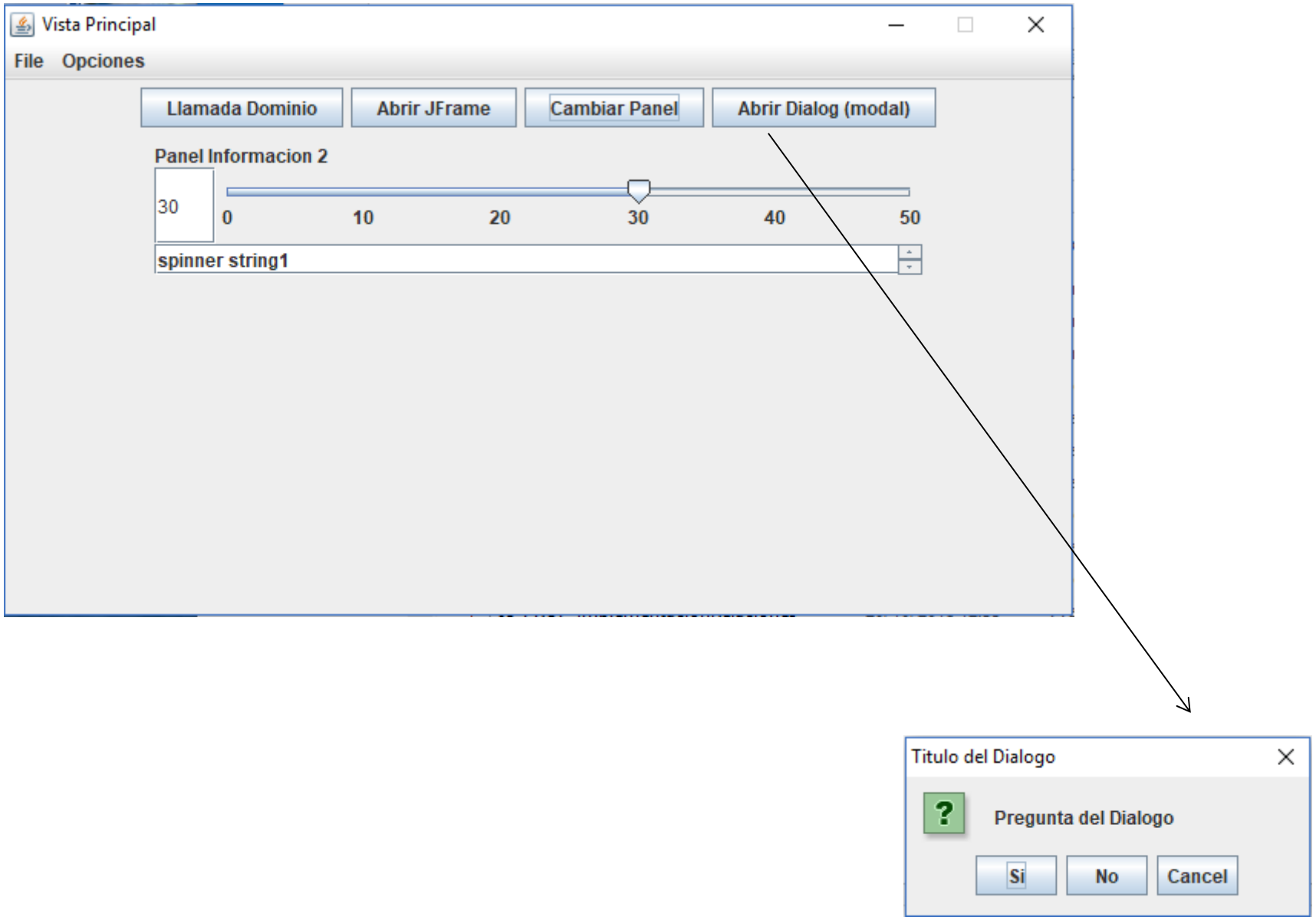

Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 1
combobox opcion1
Text Area Informacion1
Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 2
30
0
10
20
30
40
50
spinner string1


Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 1
combobox opcion1
Text Area Informacion1
combobox opcion1 resulDominio1-0
combobox opcion1 resulDominio1-1
combobox opcion1 resulDominio1-2
combobox opcion1 resulDominio1-3


Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 2
30
0
10
20
30
40
50
combobox opcion1 resulDominio1-0


Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 1
combobox opcion1
Text Area Informacion1


Vista Secundaria
Llamada Dominio
Volver
Text Area Informacion


Vista Principal
File
Opciones
Llamada Dominio
Abrir JFrame
Cambiar Panel
Abrir Dialog (modal)
Panel Informacion 2
30
0
10
20
30
40
50
spinner string1
Titulo del Dialogo
Pregunta del Dialogo
Si
No
Cancel