1) Component

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Esta clase representa un componente rectangular cualquiera de una aplicación gráfica de escritorio. Es por tanto, cualquier cosa que puede ser colocada a una ventana y con la que puede interactuar el usuario. Es la clase padre de los botones, listas, tablas, etc.

Component
+ int getX() + int getY() + int getWidth() + int getHeight() + Point getMousePosition() + boolean isVisible() + boolean isEnabled() + Color getForeground() + Color getBackground() + Font getFont() + Cursor getCursor() + void setBounds(Rectangle r) + void setVisible(boolean b) + void setEnabled (boolean r) + void setForeground(Color c) + void setBackground(Color c) + void setFont(Font f) + boolean requestFocusInWindow() + void repaint() + boolean contains(int x, int y) + Container getParent() + Component getComponentAt(int x, int y) + void add(PopupMenu menú)

- Los métodos getX y getY devuelve en valor de las coordenadas del componente.
- Los métodos getWidth y getHeight devuelven la anchura y altura del componente.
- getMousePosition devuelve el punto donde está el ratón dentro del componente.
- isVisible devuelve true si el componente se muestra en pantalla
- isEnabled devuelve true si el componente está habilitado.
- getForeground devuelve el color usado para dibujar el componente
- getBackground devuelve el color usado para dibujar el fondo del componente
- getFont devuelve el tipo de letra usado para escribir dentro del componente
- getCursor devuelve el cursor del ratón usado cuando éste entra en el componente.
- Los métodos setBounds, setVisible, setEnable, setForeground y setFont cambian los valores que se han explicado en sus respectivos métodos get.
- requestFocusInWindow hace que el componente adquiera el foco (componente activo al pulsar la tecla TAB). Si no puede obtenerlo, devuelve false.
- El método repaint hace que Java vuelva a dibujar el componente.
- El método contains devuelve true si el punto indicado está dentro del componente.
- getParent devuelve el componente padre en el cual está colocado el componente.
- En aquellos componentes que están formados por subcomponentes individuales, getComponentAt devuelve el subcomponente que está en las coordenadas indicadas.
- El método add añade un menú contextual al componente.

2) Eventos de la clase Component

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

La siguiente lista muestra los eventos que puede generar un objeto de la clase Component y que por tanto, pueden ser procesados por una aplicación mediante un manejador de eventos. Son los eventos comunes a todos los componentes de Java y están agrupados por categorías:

Categoría	Evento	Cuándo se lanza
Componente	componentHidden	El componente deja de ser visible
	componentMoved	El componente se mueve
	componentResized	El componente cambia de tamaño
	componentShown	El componente se muestra por primera vez
Foco	focusGained	El componente adquiere el foco
	focusLost	El componente pierde el foco
Teclado	keyPressed	Una tecla se pulsa sobre el componente
	keyReleased	Una tecla se suelta sobre el componente
	keyTyped	Una tecla es escrita en el componente
Ratón	mousePressed	Un botón del ratón es pulsado en el componente
	mouseReleased	Un botón del ratón se suelta en el componente
	mouseEntered	El ratón entra en el componente
	mouseExited	El ratón sale del componente
	mouseClicked	Un botón del ratón es pulsado y soltado en el componente
Movimiento	mouseMoved	El ratón se mueve dentro del componente
	mouseDragged	El ratón es arrastrado dentro del componente
Rueda	mouseWheelMoved	La rueda del ratón se mueve dentro del componente

A su vez, por cada categoría de evento hay una clase que proporciona información sobre los detalles del evento. Por ejemplo, para los eventos del ratón está la clase MouseEvent y para los eventos del teclado, KeyEvent. Con estas clases se puede obtener por ejemplo, el botón del ratón pulsado, o la tecla pulsada sobre el componente.

3) JLabel

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JLabel es una etiqueta de texto situada en un lugar del formulario. Admite un texto normal plano como rótulo, o un String en formato HTML. Por convención, el nombre de los objetos de esta clase debe empezar por **lbl**

JLabel extends Component
- String text - boolean opaque - Font font - Icon icon
+ String getText() + Icon getIcon() + boolean isOpaque() + Font getFont() + void setText(String txt) + void setOpaque(boolean b) + void setFont(Font f) + void setIcon(Icon i) + void setHorizontalAligment(int t) + void setVerticalAligment(int t)

- El método getText devuelve el texto que hay en la etiqueta
- El método getIcon devuelve el icono que hay en la etiqueta
- El método isOpaque devuelve true si la etiqueta no tiene fondo transparente. Por defecto, las etiquetas tienen fondo transparente (opaque=false).
- El método getFont devuelve el tipo de letra usado en la etiqueta
- El método setText cambia el texto visible en la etiqueta
- El método setOpaque indica si la etiqueta tendrá o no fondo transparente
- El método setFont indica el tipo de letra usado en la etiqueta.
- El método setIcon permite cambiar el icono. Puede admitir un objeto de la clase ImageIcon.
- El método setHorizontalAligment establece el tipo de alineación horizontal del texto. Debe pasarse una de estas constantes:

SwingConstants.LEFT	Texto alineado a la izquierda
SwingConstants.RIGHT	Texto alineado a la derecha
SwingConstants.CENTER	Texto alineado en la posición central del área de la etiqueta

- El método setVerticalAligment establece el tipo de alineación vertical del texto. Admite una de estas constantes:

SwingConstants.TOP	Texto alineado en la parte superior de la etiqueta
SwingConstants.CENTER	Texto alineado en la parte central de la etiqueta
SwingConstants.BOTTOM	Texto alineado en la parte inferior de la etiqueta

4) JButton

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JButton es un botón que se coloca en un formulario y puede ser pulsado. Por convención, los objetos de esta clase deben comenzar por **cmd**

JButton extends Component
- String text
+ String getText() + void setText(String t)

Propiedades destacadas

- Text: Es el texto que hay dentro del botón

Métodos destacados

- getText devuelve el texto que hay dentro del botón
- setText cambia el texto que hay dentro del botón.

Eventos destacados

- mouseClicked: se lanza cuando el botón es pulsado. La información del estado del ratón se pasa al manejador de eventos como un objeto de la clase MouseEvent:

MouseEvent
+ int getButton() + int getClickCount() + int getX() + int getY() + Point getLocationOnScreen()

- getButton devuelve una constante con el botón que ha sido pulsado:

MouseEvent.BUTTON_1	Botón izquierdo del ratón
MouseEvent.BUTTON_2	Botón derecho del ratón
MouseEvent.BUTTON_3	Botón central del ratón (si lo tiene)

- getClickCount devuelve el número de pulsaciones realizadas por el usuario
- getX devuelve la coordenada X del ratón respecto del botón.
- getY devuelve la posición del Y del ratón respecto del botón.
- getLocationOnScreen devuelve la posición del ratón respecto de la esquina superior izquierda de la pantalla.

5) JTextField

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JTextField es un cuadro de texto donde un usuario puede escribir un texto pequeño. Por convención, los objetos de esta clase deben comenzar por **txt**

JTextField extends Component
- String text - boolean editable
+ String getText() + void setText(String t) + boolean isEditable() + void setEditable (boolean b)

Propiedades destacadas

- Text: Es el texto que hay dentro del cuadro de texto
- editable: nos indica si el cuadro de texto es de solo lectura (false) o no.

Métodos destacados

- getText devuelve el texto que hay dentro del cuadro de texto
- setText cambia el texto que hay dentro del cuadro de texto.
- isEditable devuelve true si el cuadro de texto es editable
- setEditable permite poner el cuadro de texto como editable o no editable.

Eventos destacados

- keyTyped: se lanza cada vez que una letra se escribe en el cuadro de texto. Este evento produce un objeto de la clase KeyEvent:

KeyEvent
+ int getKeyCode() + char getKeyChar() + boolean isAltDown() + boolean isControlDown() + boolean isShiftDown()

- El método getKeyCode devuelve una constante con el identificador de la tecla que ha sido pulsada.
- El método getKeyChar devuelve un char con la tecla pulsada, si ésta corresponde a una tecla. Hay teclas (como control, mayúsculas, etc) que no lo proporcionan.
- isAltDown devuelve true si la tecla alt está pulsada en ese momento
- isControlDown devuelve true si la tecla control está pulsada en ese momento
- isShiftDown devuelve true si la tecla mayúsculas está pulsada en ese momento.

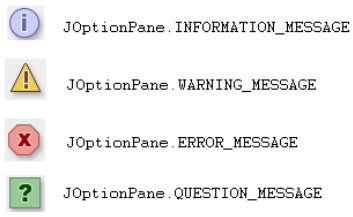
6) JOptionPane

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Esta clase permite mostrar todo tipo de ventanas emergentes en la pantalla.

JOptionPane
+ static void showMessageDialog(Component p, String mensaje, String titulo, int tipo) + static String showInputDialog(Component p, String mensaje, String titulo, int tipo) + static int showConfirmDialog(Component p, String mensaje, String titulo, int botones, int tipo)

- **showMessageDialog:** Crea una ventana emergente de información. Debe pasarse en primer lugar el componente padre de la ventana (o null), el texto del mensaje, el título de la ventana emergente y una de las siguientes constantes para definir su icono:



- **showInputDialog:** Crea una ventana emergente en la que se pide una frase al usuario. Debe pasarse en primer lugar el componente padre de la ventana (o null), un texto con un mensaje (normalmente será una pregunta dirigida al usuario), el título de la ventana y una constante de las anteriores para el icono. El método retorna un String con la frase escrita por el usuario.

- **showConfirmDialog:** Crea una ventana emergente con un mensaje y varios botones, de forma que el usuario puede elegir entre varias opciones. Al igual que las anteriores, recibe el componente padre (o null), textos del mensaje, título de la ventana, una constante que indica los botones que aparecen y una constante con el tipo de icono.

Las constantes que indican los botones que deben mostrarse son:

JOptionPane.YES_NO_OPTION	Muestra los botones “Si” y “No”
JOptionPane.YES_NO_CANCEL_OPTION	Muestra los botones “Si”, “No” y “Cancelar”
JOptionPane.OK_CANCEL_OPTION	Muestra los botones “Aceptar” y “Cancelar”

El método devuelve una constante con lo que ha hecho el usuario:

JOptionPane.YES_OPTION	Se ha pulsado el botón “Si”
JOptionPane.NO_OPTION	Se ha pulsado el botón “No”
JOptionPane.OK_OPTION	Se ha pulsado el botón “Aceptar”
JOptionPane.CANCEL_OPTION	Se ha pulsado el botón “Cancelar”
JOptionPane.CLOSED_OPTION	El usuario ha cerrado la ventana sin elegir nada

7) JCheckBox

- **Librería:** *Swing*
- **Desarrollador:** *Sun Microsystems*
- **Archivo:** *Incluido en el JDK*
- **Paquete:** *javax.swing*

Un JCheckBox es una casilla de verificación. Por convenio su nombre debe comenzar por **chk**

JCheckBox extends Component
- String text - boolean selected - boolean enabled
+ boolean isSelected() + void setSelected(boolean b) + void setText(String t) + String getText() + void setEnabled(boolean b) + boolean isEnabled()

Propiedades destacadas

- text: Es el texto que acompaña la casilla de verificación.
- selected: Si es true, la casilla está marcada.
- enabled: Si es true, la casilla está habilitada

Métodos destacados

- isSelected devuelve true si la casilla está marcada
- setSelected activa o desactiva la casilla de verificación
- getText devuelve el texto que acompaña la casilla de verificación
- setText cambia el texto que acompaña la casilla de verificación.
- setEnabled habilita o deshabilita la casilla de verificación
- isEnabled devuelve true si la casilla de verificación está habilitada.

Eventos destacados

- itemStateChange: se lanza cada vez que se activa o desactiva la casilla de verificación. La información de lo que sucede se pasa al manejador de eventos como un objeto de la clase ItemEvent

ItemEvent
+ void getStateChange()

- getStateChange: Devuelve el estado al que pasa la casilla de verificación, que es una de estas constantes:

ItemEvent.SELECTED	La casilla pasa a estar marcada
ItemEvent.DESELECTED	La casilla ha sido desmarcada.

8) JRadioButton

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JRadioButton es un botón de radio que puede ser seleccionado por el usuario. Todos aquellos botones de radio que posean el mismo ButtonGroup son excluyentes. Por convenio, los nombres de los botones de radio deben comenzar por **opt.**

JRadioButton extends Component
- String text - boolean selected - boolean enabled - ButtonGroup buttonGroup
+ void setButtonGroup(ButtonGroup g) + boolean isSelected() + void setSelected(boolean b) + void setText(String t) + String getText() + void setEnabled(boolean b) + boolean isEnabled()

Propiedades destacadas

- buttonGroup: Es un objeto que indica un grupo excluyente de botones de radio.
- text: Es el texto que acompaña la casilla de verificación.
- selected: Si es true, la casilla está marcada.
- enabled: Si es true, la casilla está habilitada

Métodos destacados

- isSelected devuelve true si la casilla está marcada
- setSelected activa o desactiva la casilla de verificación
- getText devuelve el texto que acompaña la casilla de verificación
- setText cambia el texto que acompaña la casilla de verificación.
- setEnabled habilita o deshabilita la casilla de verificación
- isEnabled devuelve true si la casilla de verificación está habilitada.

Eventos destacados

- itemStateChange: se lanza cada vez que se activa o desactiva la casilla de verificación. La información de lo que sucede se pasa al manejador de eventos como un objeto de la clase ItemEvent, que hemos visto en los JCheckBox.

9) JFrame

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JFrame es un formulario, es decir, la ventana principal de una aplicación de escritorio. Por convenio, su nombre debe comenzar por **frm**.

JFrame extends Component
- int defaultCloseOperation - boolean allwaysOnTop - boolean resizable - String title - boolean undecorate
+ void dispose() + Point getMousePosition() + Graphics getGraphics() + void setIcon(Image img) + void setLocationRelativeTo(Component c)

Propiedades destacadas

- defaultCloseOperation: constante que indica lo que sucede si se pulsa la "X" de cerrar:

JFrame.EXIT_ON_CLOSE	La aplicación entera finaliza cuando se cierra el JFrame
JFrame.DISPOSE	La ventana se cierra, pero no la aplicación, que sigue funcionando
JFrame.DO_NOTHING	No pasa nada al pulsar la X. El programador debe cerrar la ventana usando el método dispose

- allwaysOnTop: booleano que indica si la ventana está siempre visible
- resizable: booleano que indica si la ventana puede ser cambiada de tamaño
- title: Título que aparece en la parte superior azul de la ventana.
- undecorate: booleano que indica si la ventana debe tener la parte superior azul.

Métodos destacados

- dispose cierra la ventana, aunque no se haya pulsado la X de cerrar.
- getMousePosition devuelve la posición del ratón dentro de la ventana.
- getGraphics devuelve un objeto para dibujar sobre la ventana. No obstante, hay que usarlo con precaución ya que el sistema operativo puede redibujar la ventana en cualquier momento y eliminar lo que haya sido dibujado con él.
- setIcon cambia el icono de la ventana.
- setLocationRelativeTo indica el componente respeto al cual debe aparecer centrado el formulario. Si se pone null, se centra en la pantalla.

Eventos destacados

- windowOpened: Se lanza al abrir por primera vez el formulario.
- windowClosing: Se lanza al pulsar la X de cerrar, pero con el formulario aún abierto. Sirve para prevenir que el usuario pueda cerrarlo accidentalmente.
- windowClosed: Se lanza cuando el formulario se ha cerrado definitivamente.
- windowStateChanged: Se lanza al maximizar o minimizar el formulario.

10) JFileChooser

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JFileChooser es un objeto que muestra la ventana de selección de archivos del sistema operativo. Este objeto solo sirve para seleccionar archivos y directorios, y en ningún caso se realiza ningún procesado con el archivo seleccionado, esto ya correspondería al programa.

JFileChooser
+ JFileChooser() + JFileChooser(File directorio) + int showOpenDialog(Component padre) + int showSaveDialog(Component padre) + File getSelectedFile() + File[] getSelectedFiles() + File getCurrentDirectory() + void setFileSelectionMode(int modo) + void setMultiSelectionEnabled(boolean b) + void setFileFilter (FileFilter f)

- El primer constructor crea un cuadro de selección de archivos en el directorio principal asignado al usuario por el sistema operativo.
- El segundo constructor crea un cuadro de selección de archivos en el directorio indicado como parámetro.
- showOpenDialog abre un cuadro de selección de archivos para abrir un archivo o directorio. Es centrado en el componente pasado como parámetro. Si se pone null, se centra automáticamente en la pantalla.
- showSaveDialog es como el anterior, pero para guardar archivos.
- getSelectedFile devuelve el archivo o directorio seleccionado por el usuario
- getSelectedFiles devuelve una lista con todos los archivos y directorios seleccionados
- getCurrentDirectory devuelve el directorio actual del cuadro de selección de archivos.
- setFileSelectionMode recibe una constante que indica aquello que puede ser seleccionado:

JFileChooser.FILES_ONLY	Solo se pueden seleccionar archivos
ListSelectionModel.DIRECTORIES_ONLY	Solo se pueden seleccionar directorios
ListSelectionModel.FILES_AND_DIRECTORIES	Solo se pueden seleccionar archivos y directorios

- setMultiSelectionEnabled permite indicar si se pueden elegir una o varias cosas
- setFileFilter indica los archivos admitidos. Debe recibir un objeto de una clase hija de FileFilter que implemente el filtro correspondiente.

abstract FileFilter
+ abstract boolean accept (File f) + abstract String getDescription()

11) UIManager

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Esta clase permite cambiar la apariencia (Look and Feel) de una aplicación de escritorio.

UIManager
+ static void setLookAndFeel(String identificador) throws Exception + static String getSystemLookAndFeelClassName() + static String getCrossPlatformLookAndFeelClassName()

- El método setLookAndFeel cambia la apariencia de la aplicación. Recibe como parámetro el identificador del Look and Feel que vamos a poner. Antes de llamar a este método hay que agregar al proyecto la librería de ese Look And Feel.

  Con Java vienen cuatro apariencias (en algunos sistemas operativos como Mac OS puede haber más) que pueden utilizarse siempre:

Nombre	Identificador	Descripción
Metal	javax.swing.plaf.metal.MetalLookAndFeel	Apariencia clásica de Java hasta la versión 7
Motif	com.sun.java.swing.plaf.motif.MotifLookAndFeel	Apariencia de los primeros sistemas Linux
Windows	com.sun.java.swing.plaf.windows.WindowsLookAndFeel	Apariencia de aplicación Windows
Nimbus	javax.swing.plaf.nimbus.NimbusLookAndFeel	Apariencia mejorada (a partir de Java 7)

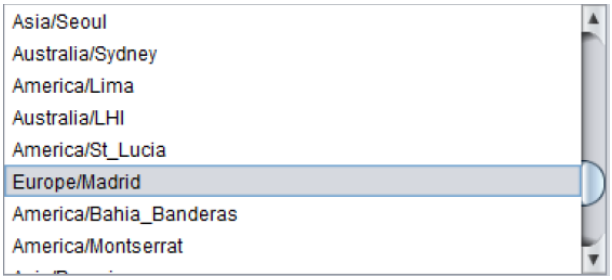
- El método getSystemLookAndFeelClassName devuelve el identificador del Look And Feel que más se ajuste al sistema operativo utilizado. Esto hace que la apariencia de la aplicación se parezca a las programadas en otros lenguajes específicos de la plataforma.

- El método getCrossPlatformLookAndFeelClassName devuelve el identificador de la apariencia "Metal", que es la que tenían todas las aplicaciones de Java hasta la versión 7 independientemente del sistema operativo utilizado.

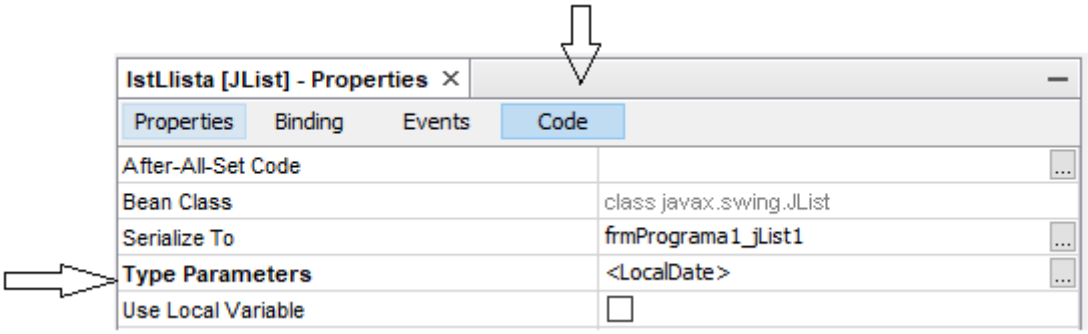
12) JList<T>

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JList<T> es un componente de la interfaz de usuario que muestra una lista de objetos de tipo T para que el usuario elija uno de ellos. Su nombre debe comenzar por **lst**.



Cuando arrastramos el JList<T> a la interfaz lo primero que debemos hacer (además de cambiarle el nombre de variable) es dar forma a su generic para indicar el tipo de dato de los objetos que va a contener. Eso se hace pulsando el botón "Code" que hay en la zona de propiedades y escribiendo en "type parameters" el generic. Por ejemplo, si queremos guardar en la lista objetos de tipo LocalDate, escribiremos esto:



El JList<T> en realidad no contiene objetos, sino que su función es **visualizar** objetos, llamando al método **toString** de cada uno. Los objetos deben encontrarse en un objeto de tipo **DefaultListModel<T>**, que es una estructura de datos similar a una lista. (Guardar los objetos en un sitio y usar un objeto diferente para mostrarlos se llama **Modelo Vista Controlador**).

JList<T>
+ void setModel(DefaultListModel<T> listaObjetos) + T getSelectedValue() + T[] getSelectedValues()

- **setModel:** Hace que el JList<T> muestre los objetos de la lista que se pasa como parámetro.
- **getSelectedValue:** Devuelve el objeto que está seleccionado en el JList.
- **getSelectedValues:** Devuelve una lista con todos los objetos seleccionados en el JList

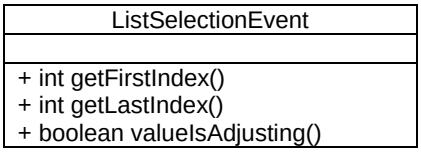
Propiedades destacadas

- **selectionMode**: Constante que indica cuántos objetos se pueden seleccionar a la vez:

ListSelectionModel.SINGLE_SELECTION	Solo puede estar seleccionado un objeto en la lista
ListSelectionModel.SINGLE_INTERVAL_SELECTION	Puede seleccionarse un intervalo en la lista
ListSelectionModel.MULTIPLE_INTERVAL_SELECTION	Pueden seleccionarse varios objetos en la lista

Eventos destacados

- **valueChange**: Se lanza cuando cambia el valor seleccionado en la lista. La información se manda como un objeto de la clase ListSelectionEvent:



  - **getFirstIndex:** devuelve el primer valor del rango seleccionado.
  - **getLastIndex:** devuelve el último valor del rango seleccionado.
  - **valueIsAdjusting:** devuelve true si el valor de la selección está todavía cambiando debido a la acción realizada por el usuario y aún no está estabilizado. Debería comprobarse esto antes de usar los métodos getFirstIndex y getLastIndex.

13) DefaultListModel<T>

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Esta clase es una estructura de datos similar a una lista (secuencia de datos ordenados en la que cada uno tiene una posición). Se utiliza para guardar los objetos que posteriormente serán visualizados en un JList<T>. Sus métodos son casi iguales a los de List<T>.

DefaultListModel<T>
+ DefaultListModel<T>() + void addElement(T objeto) + void add(int pos, T objeto) + void set (Object objeto, int posición) + T get(int posición) + int size() + void clear() + boolean contains (T objeto) + int indexOf(T objeto) + void remove(int posición) + void remove (T objeto)

- El constructor crea una lista vacía.
- El método addElement añade un objeto de tipo T al final de la lista.
- El método add inserta un objeto de tipo T en la posición de la lista pasada como primer parámetro, desplazando los elementos afectados hacia la derecha.
- El método set permite cambiar el elemento de la lista que ocupa una posición. No se puede pasar una posición que supere el tamaño de la lista.
- El método get devuelve el objeto de la lista que se encuentra en la posición especificada.
- El método size devuelve el número de objetos que hay en la lista.
- El método clear vacía la lista.
- El método contains devuelve true si el objeto pasado como parámetro está en la lista.
- El método indexOf devuelve el número de posición en la que se encuentra el objeto pasado como parámetro. Si el objeto no se encuentra en la lista, devuelve -1.
- El primer método remove borra de la lista el objeto cuya posición se pasa como parámetro, desplazando los objetos restantes una posición hacia la izquierda.
- El segundo método remove borra de la lista la primera aparición del objeto pasado como parámetro. Si no está en la lista, no hace nada.

14) JTable

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JTable es un componente de la interfaz de usuario que muestra una tabla cuyas celdas están rellenadas con objetos (por defecto, todos son Object, aunque se puede cambiar desde el diseñador). Su nombre debe comenzar por **tab**.

Fecha de registro	Fecha de cumplimiento	Descripción
01/01/2019 10:30	10/01/2019 19:00	Recuperación de programación
05/01/2019 15:45	17/01/2019 16:00	Comenzar tema 6 de programación
05/01/2019 15:46	17/01/2019 19:00	Comentar tema 5 de entornos
10/01/2019 12:32	24/01/2019 16:00	Recuperación de entornos

La tabla no almacena los datos. Estos se encuentran guardados en un objeto de la clase **DefaultTableModel** al que es posible acceder usando el método getModel de la tabla y **haciendo un casting a DefaultTableModel**:

```
DefaultTableModel mo=(DefaultTableModel)tabAlumnos.getModel();
```

JTable
+ int getRowCount() + int getColumnCount() + Object getValueAt(int fila, int columna) + void setValueAt(Object o, int fila, int columna) + TableModel getModel() + int[] getSelectedRows() + Vector getDataVector()

- **getRowCount:** Devuelve el número de filas de la tabla
- **getColumnCount:** Devuelve el número de columnas de la tabla
- **getValueAt:** Devuelve el objeto que se encuentra en la fila y columnas indicada. Como lo devuelve en forma de Object, lo más normal es que sea necesario hacerle un casting para recuperar el tipo de dato original.
- **setValueAt:** Actualiza una fila y columna de la tabla con el objeto que se pasa como parámetro. Es necesario que dicha fila y columna existan en la tabla, porque si no se lanzará un IndexOutOfBoundsException. Por tanto, este método no puede usarse para añadir una nueva fila a la tabla, sino para actualizar lo que ya hay en ella.
- **getModel:** Devuelve el objeto que contiene la colección de filas de la tabla. Es necesario hacerle un casting a DefaultTableModel.
- **getSelectedRows:** Devuelve un array con los números (comenzando en 0) de las filas que el usuario ha seleccionado en la tabla.
- **getDataVector:** Devuelve una lista (Vector, que es una clase antigua que implementa List) con todas las filas de la tabla.

15) DefaultTableModel

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un DefaultTableModel es una estructura de datos que guarda los datos que se mostrarán en un JTable. Internamente almacena la secuencia de filas de una tabla, siendo cada fila un array de Object.

DefaultTableModel
+ void addRow(Object[] fila) + Vector getDataVector()

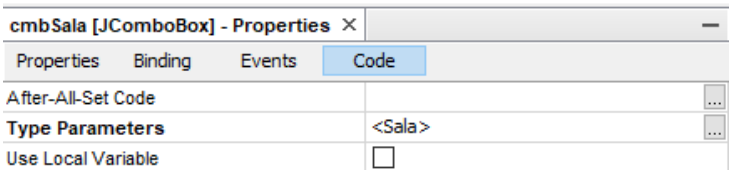
16) JComboBox<T>

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Un JComboBox<T> es una lista desplegable que contiene objetos dentro y podemos seleccionar uno de ellos. Dentro de él podemos ver lo que produce la salida del método toString de dichos objetos. Por convenio, el nombre del JComboBox<T> comienza por **cmd**.

JComboBox<T>
+ void addItem(T o) + T getSelectedItem() + int getItemCount() + void removeItemAt(int n) + void removeAllItems()

Cuando arrastramos el JComboBox<T> a la interfaz lo primero que debemos hacer (además de cambiarle el nombre de variable) es dar forma a su generic para indicar el tipo de dato de los objetos que va a contener. Eso se hace pulsando el botón “Code” que hay en la zona de propiedades y escribiendo en “type parameters” el generic. Ejemplo:



- **addItem:** Añade un objeto de tipo T al JComboBox<T>
- **getSelectedItem:** Devuelve el objeto que está seleccionado en el JComboBox<T>
- **getItemCount:** Devuelve el número total de objetos que hay en el JComboBox<T>
- **removeElementAt:** Elimina el objeto que se encuentra en la posición indicada
- **removeAllItems:** Borra todos los objetos que guarda el JComboBox<T>

Evento destacado

- **itemStateChange**: Se lanza cuando se selecciona o deja de estar seleccionado un objeto de la lista. La información se envía al manejador de eventos como un objeto de la clase **ItemEvent**

17) ItemEvent

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Cuando cambia el objeto seleccionado en un JComboBox<T>, se lanzan dos eventos de tipo itemStateChange.

- Uno para notificar que el objeto que estaba seleccionado ha dejado de estarlo
- Otro para notificar que hay un nuevo objeto que pasa a estar seleccionado.

La clase ItemEvent contiene información sobre lo que ha sucedido en el evento itemStateChange. Java automáticamente proporciona un objeto (ya creado) de la clase ItemEvent como parámetro del manejador del evento itemStateChange.

ItemEvent
+ int getStateChange()

- **getStateChange:** Devuelve un número entero que hay que comparar con la siguiente constante, para saber si el evento se ha lanzado porque un objeto se ha seleccionado, o porque ha dejado de estarlo.

ItemEvent.SELECTED	El evento se debe a que un objeto ha sido seleccionado en la lista.

18) JSpinner

- **Librería:** *Swing*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.swing*

Es un control que permite a un usuario seleccionar un número entero comprendido entre un rango que podemos definir en sus propiedades. Por convenio, su nombre comenzará con las siglas **num**.

JSpinner
+ Object getValue()

- **getValue:** Devuelve un Object con el número que hay seleccionado en el JSpinner, y por tanto, deberá hacerse un casting a Integer para poder recuperarlo.