



UNIVERSIDAD DE ALMERÍA

Grado en Ingeniería Informática

Metodología de la programación

2013



Interfaces gráficas de usuario.

Programación dirigida por eventos

Java GUI's (Graphical User Interfaces)

Java JFC/Swing API

La clase Graphics

Contenedores

Layout Managers (Administradores de Presentación)

Componentes básicos

Programación dirigida por eventos (event-driven)

La programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos (eventos) que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

El programador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador de evento

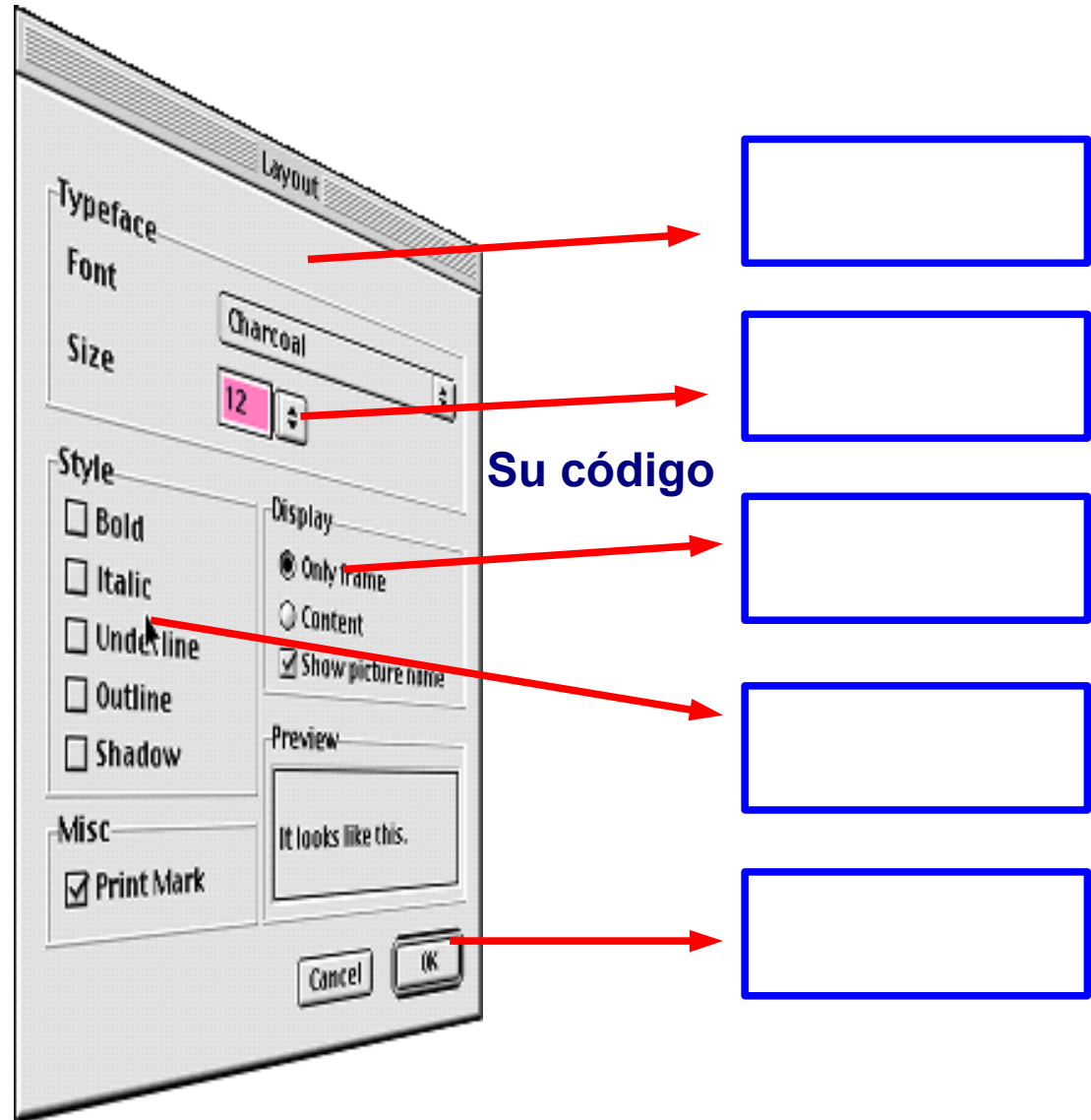
En contraposición al modelo clásico, la programación orientada a eventos permite interactuar con el usuario en cualquier momento de la ejecución. Esto se consigue debido a que los programas creados bajo esta arquitectura se componen por un bucle exterior permanente encargado de recoger los eventos, y distintos procesos que se encargan de tratarlos

La programación orientada a eventos supone una complicación añadida con respecto a otros paradigmas de programación, debido a que el flujo de ejecución del software escapa al control del programador

Los programas con GUI (Graphical User Interface) son “event-driven”

Programación dirigida por eventos

Código del sistema



Java GUI's

Para aplicaciones de escritorio (Desktop) en Java, los programadores tienen dos opciones principales.

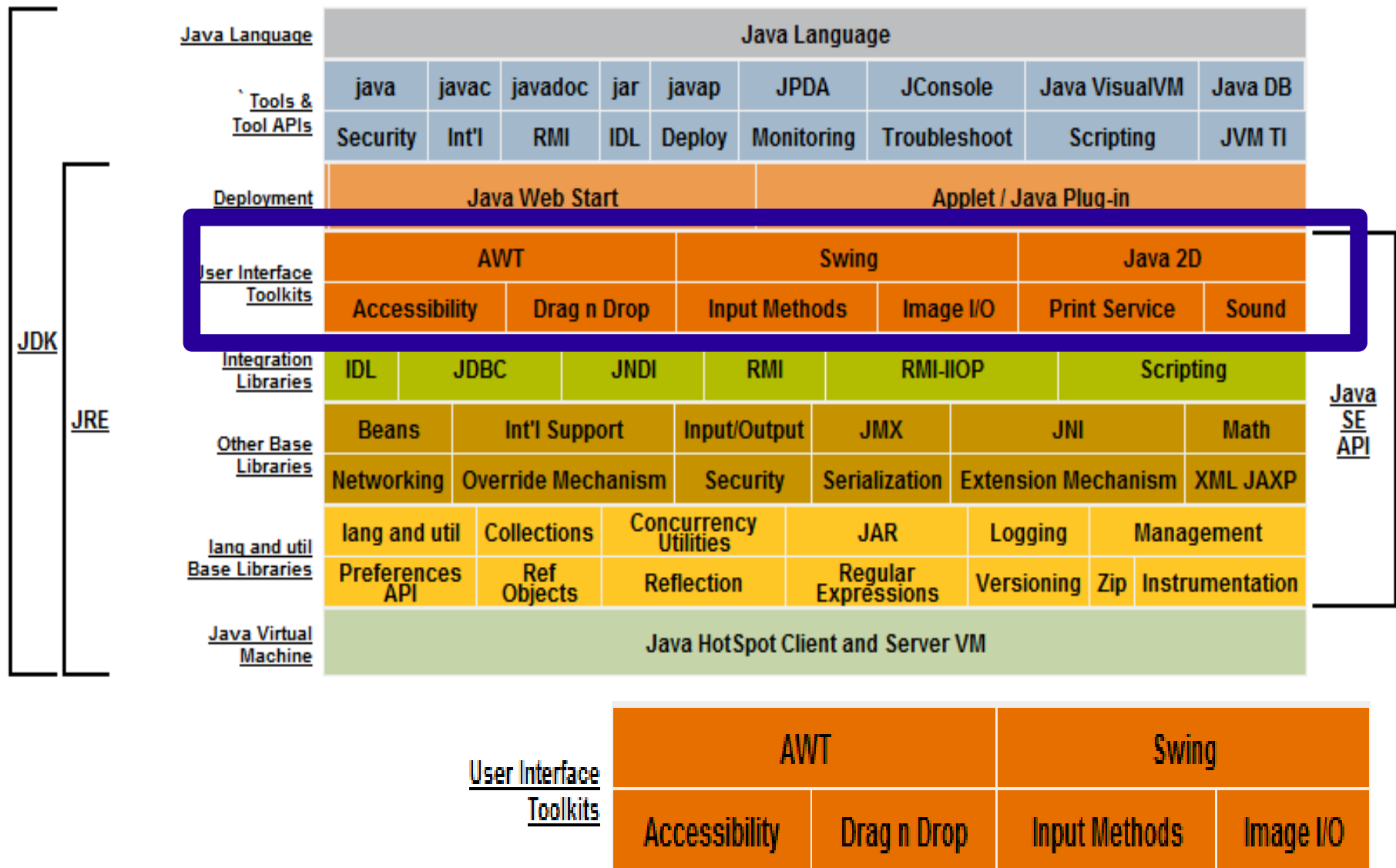
Utilizar Java Swing, integrado en el JDK,

O, utilizar el Standard Widget Toolkit (SWT) de Eclipse.

Ambos enfoques comparten algo en común, pero cada uno tiene sus propias ventajas.

Java Swing es (WORE)
“write once, run everywhere”

Java Platform Standard Edition



Java JFC/Swing GUI

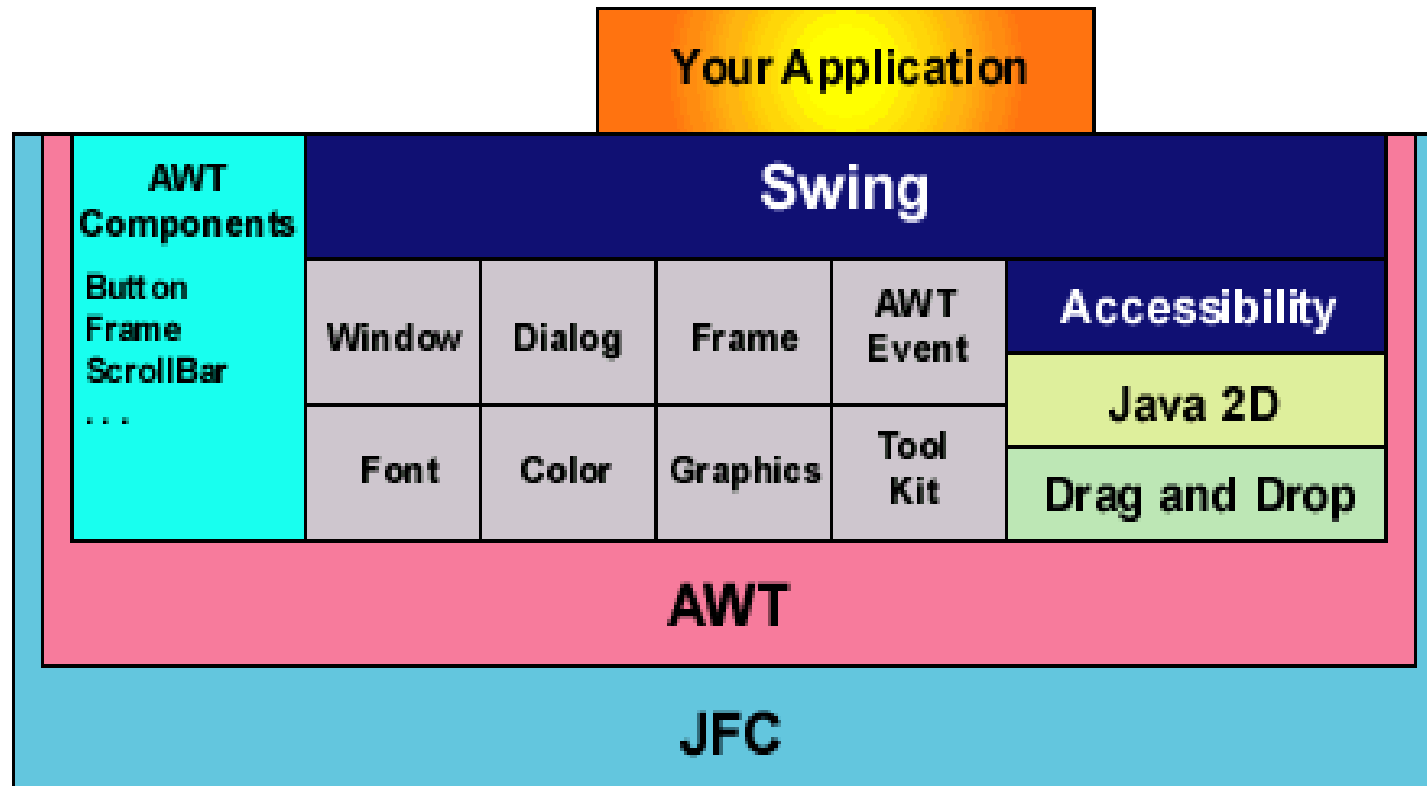
Java Foundation Classes

JFC es una colección muy grande de software.

Java provee dos librerías para crear GUI's:

Java AWT (Abstract Window Toolkit)

Java Foundation Classes (JFC o Swing), a partir de Java2



Swing/AWT

Cuando se introdujo en Java, las clases de interfaz gráfica de usuario se agrupan en una librería conocida como AWT.

AWT es útil para el desarrollo de sencillas interfaces gráficas de usuario, pero no para el desarrollo de complejos proyectos. Además, AWT es propenso a errores específicos de la plataforma.

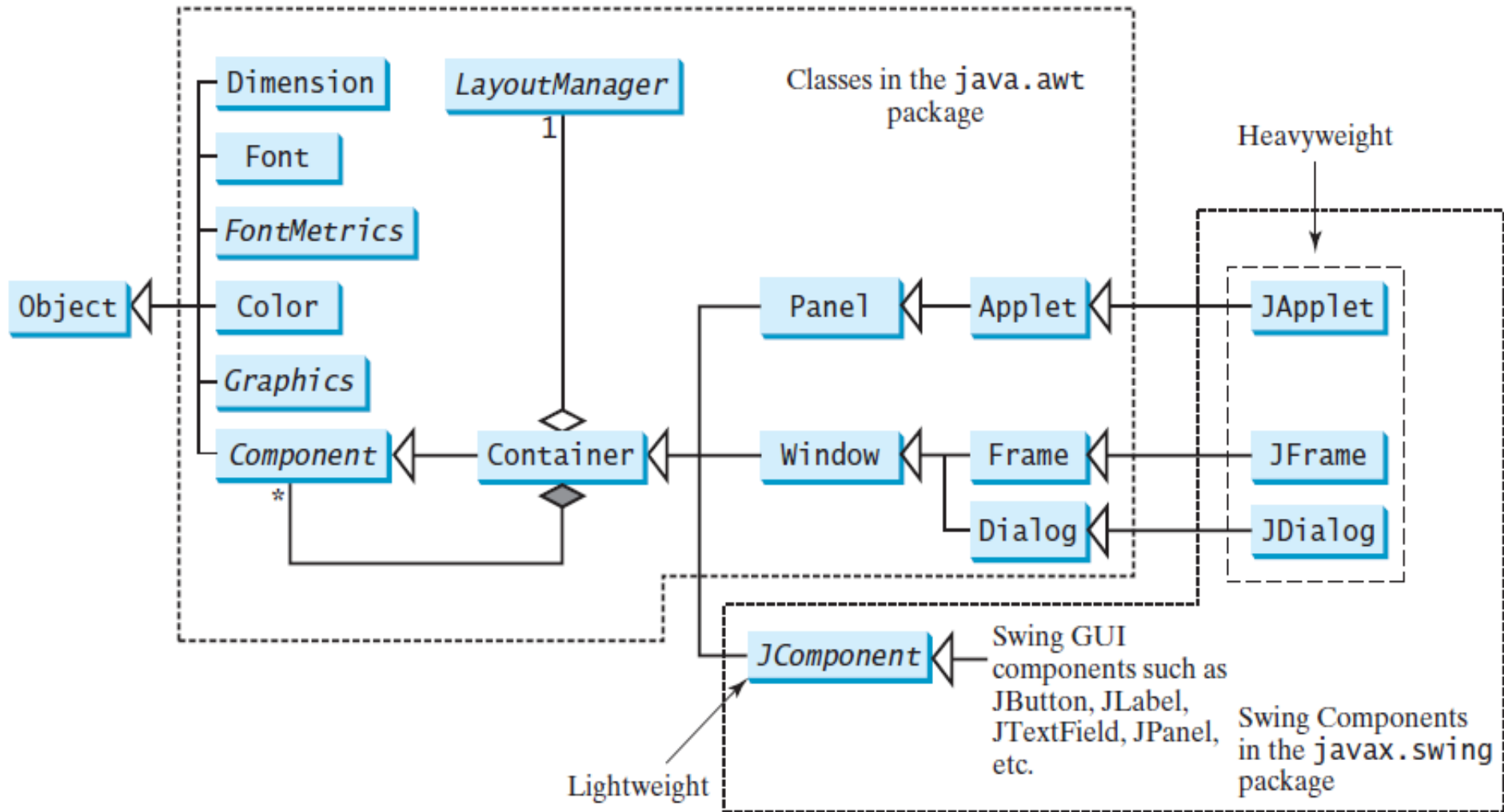
La AWT fue sustituida por una más robusta, una librería versátil y flexible conocida como Swing.

Para distinguir las nuevas clases de componentes de Swing de sus homólogos del AWT, las clases de componentes Swing se nombran con un prefijo **J**

Swing está implementada sin usar código nativo (100% Java), con lo cual la GUI se verá de la misma forma en distintas plataformas.

Los componentes Swing se denominan componentes ligeros y los componentes AWT se conocen como componentes pesados

Java GUI API



Interfaces gráficas de usuario.

Programación dirigida por eventos

Java GUI's (Graphical User Interfaces)

Java JFC/Swing API

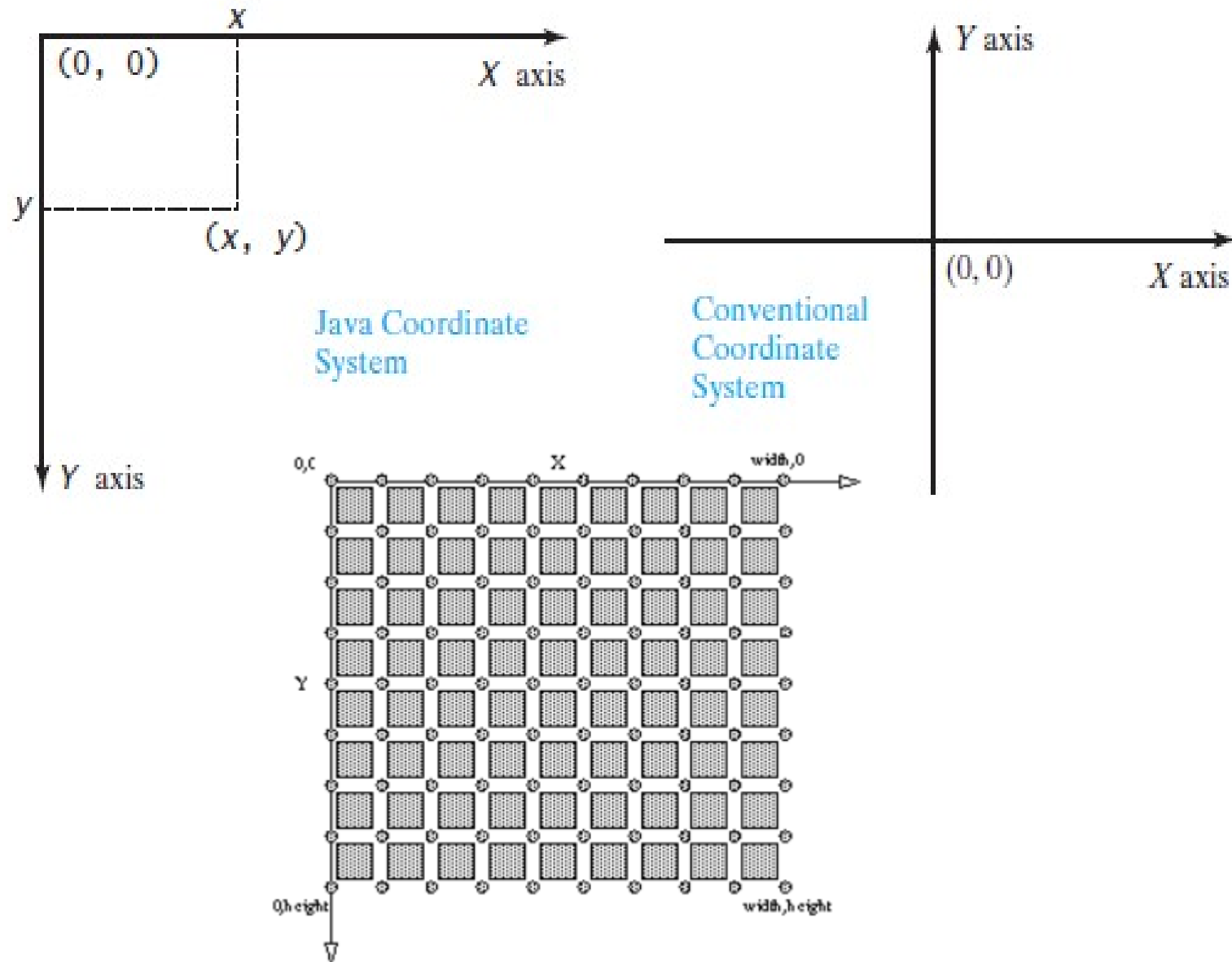
 **La clase Graphics**

Contenedores

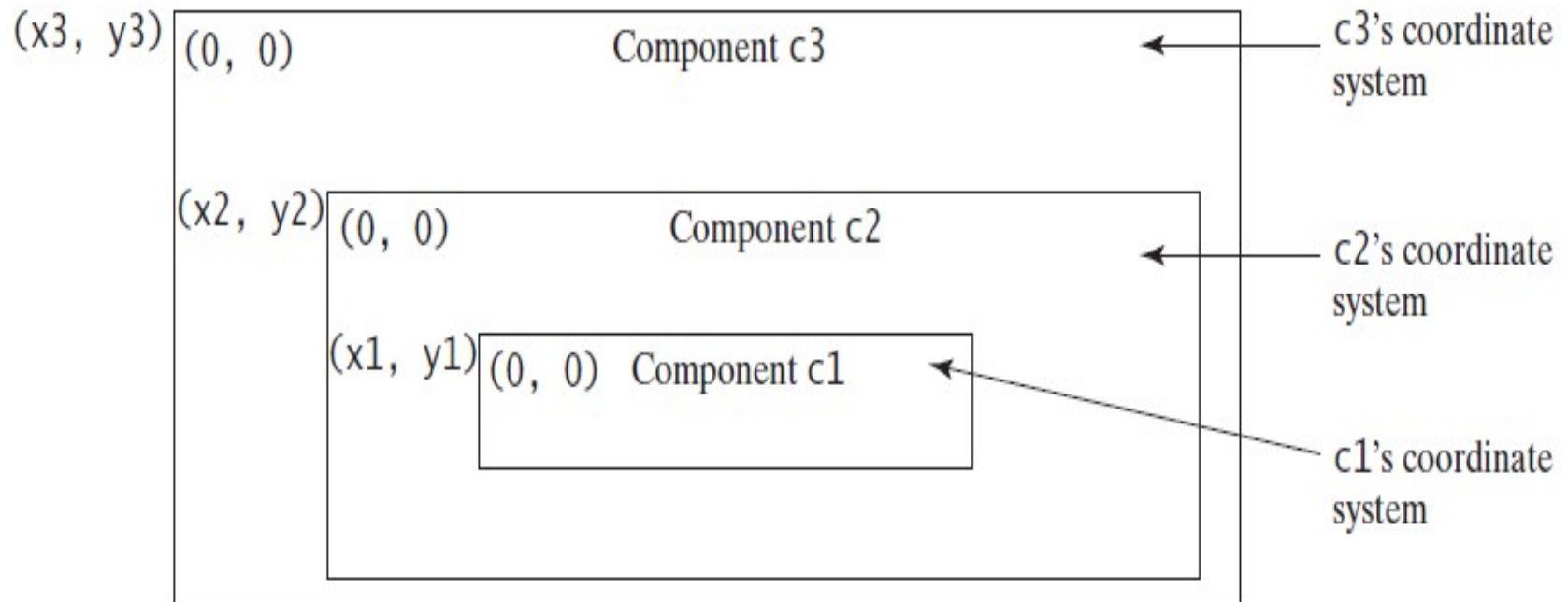
Layout Managers (Administradores de Presentación)

Componentes básicos

Sistema de coordenadas gráfico



Cada componente tiene su propio sistema de coordenadas



$(x1, y1) = (c1.getX(), c1.getY())$

$(x2, y2) = (c2.getX(), c2.getY())$

$(x3, y3) = (c3.getX(), c3.getY())$

La clase Graphics



```
java.awt.Graphics

+setColor(color: Color): void
+setFont(font: Font): void
+drawString(s: String, x: int, y: int): void
+drawLine(x1: int, y1: int, x2: int, y2:
    int): void
+drawRect(x: int, y: int, w: int, h: int):
    void
+fillRect(x: int, y: int, w: int, h: int): void

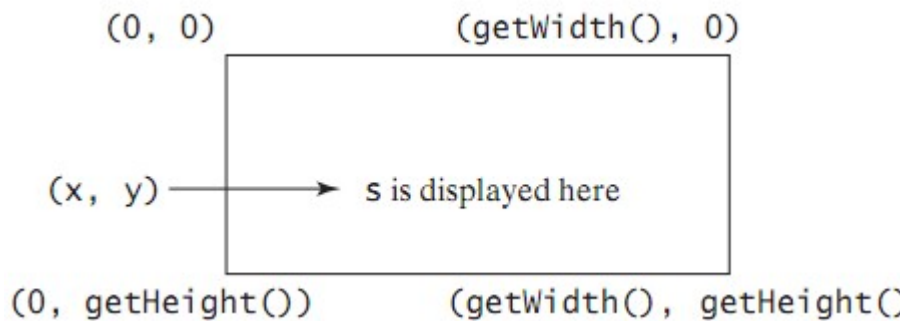
+drawRoundRect(x: int, y: int, w: int, h: int, aw:
    int, ah: int): void
+fillRoundRect(x: int, y: int, w: int, h: int,
    aw: int, ah: int): void
+draw3DRect(x: int, y: int, w: int, h: int,
    raised: boolean): void
+fill3DRect(x: int, y: int, w: int, h: int,
    raised: boolean): void
+drawOval(x: int, y: int, w: int, h: int):
    void
+fillOval(x: int, y: int, w: int, h: int): void

+drawArc(x: int, y: int, w: int, h: int,
    startAngle: int, arcAngle: int): void
+fillArc(x: int, y: int, w: int, h: int,
    startAngle: int, arcAngle: int): void
+drawPolygon(xPoints: int[], yPoints:
    int[], nPoints: int): void
+fillPolygon(xPoints: int[], yPoints: int[],
    nPoints: int): void
+drawPolygon(g: Polygon): void
+fillPolygon(g: Polygon): void
+drawPolyline(xPoints: int[], yPoints:
    int[], nPoints: int): void
```

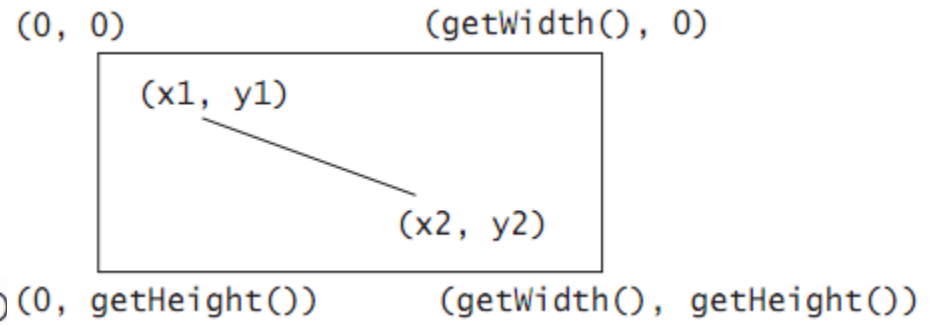
La clase Graphics contiene los métodos para dibujar texto y figuras geométricas

Dibujo de cadenas (Strings), líneas, rectángulos y óvalos

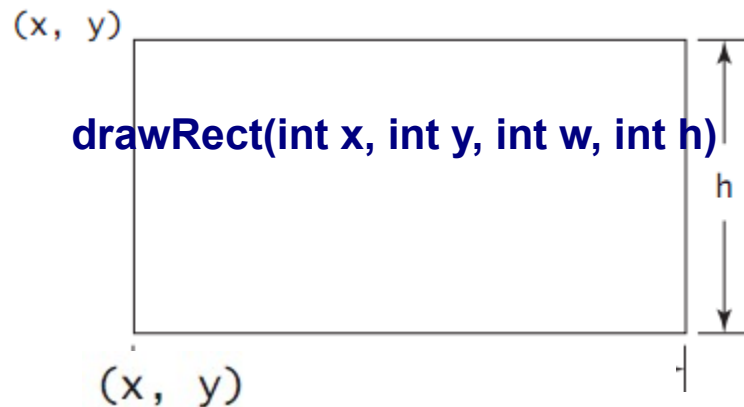
drawString(s,x,y)



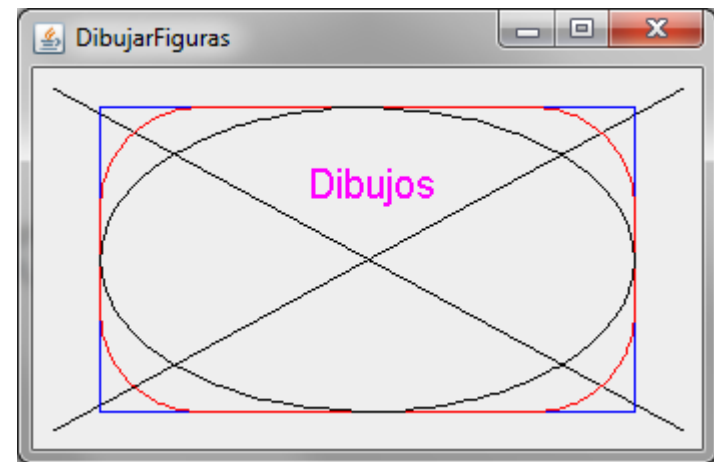
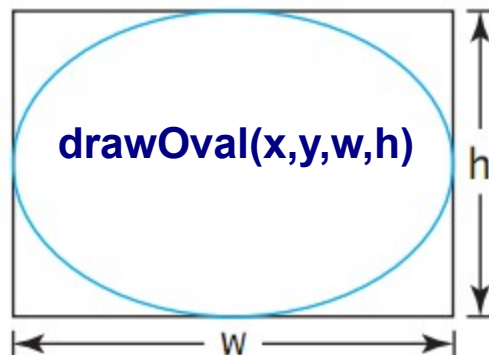
drawLine(int x1, int y1, int x2, int y2)



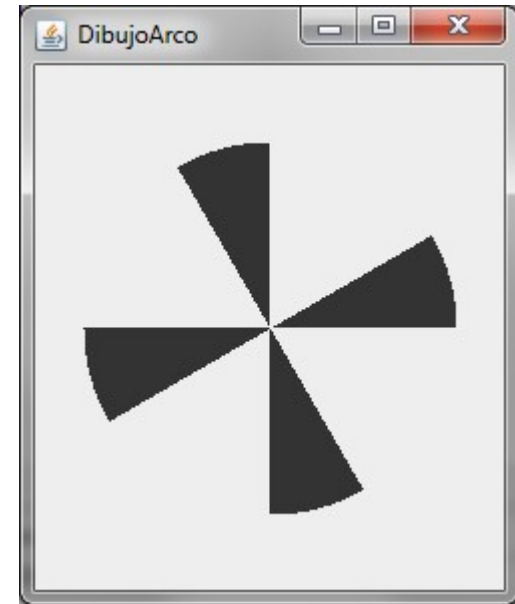
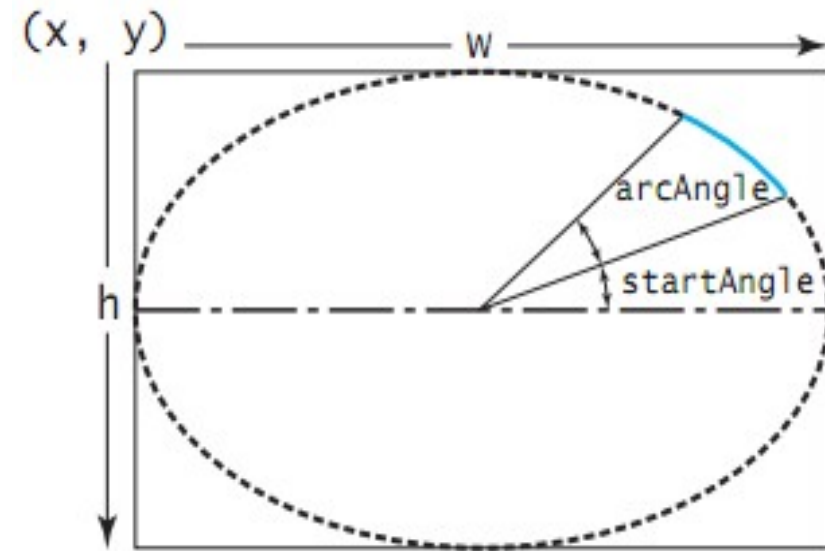
drawRect(int x, int y, int w, int h)



drawOval(x,y,w,h)



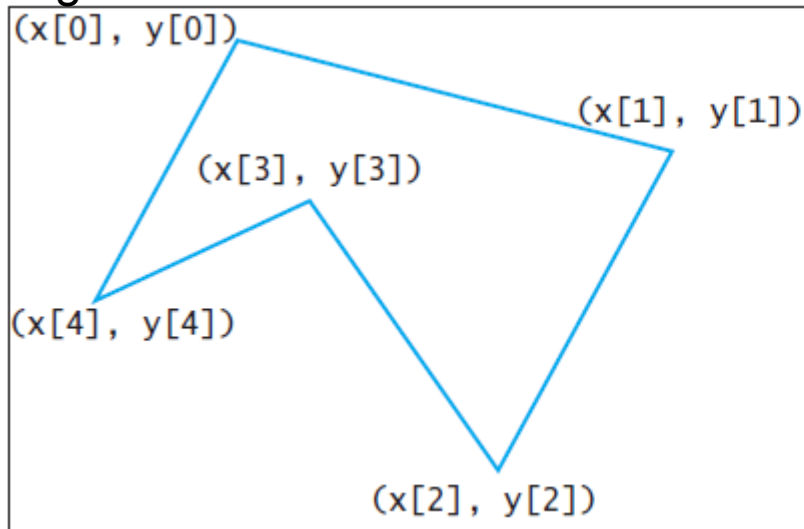
Dibujar un arco



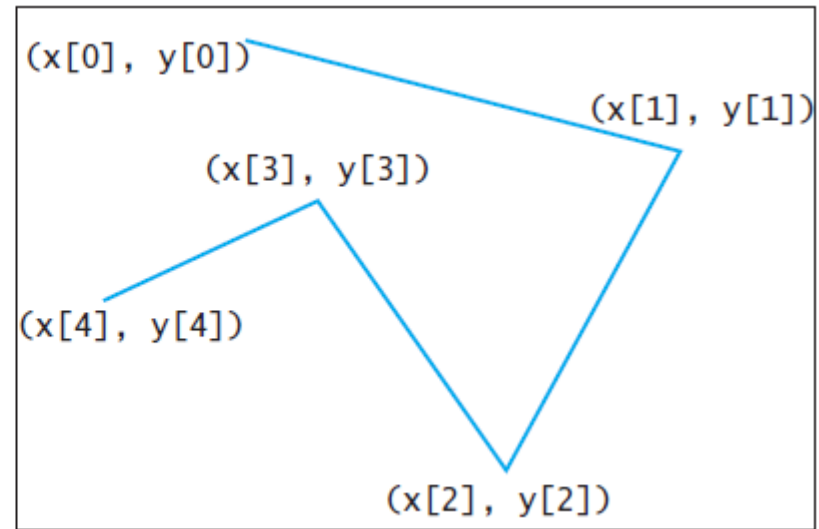
```
drawArc(int x, int y, int w, int h, int startAngle, int arcAngle);  
fillArc(int x, int y, int w, int h, int startAngle, int arcAngle);
```

Dibujar polígonos y polilíneas

Polígono



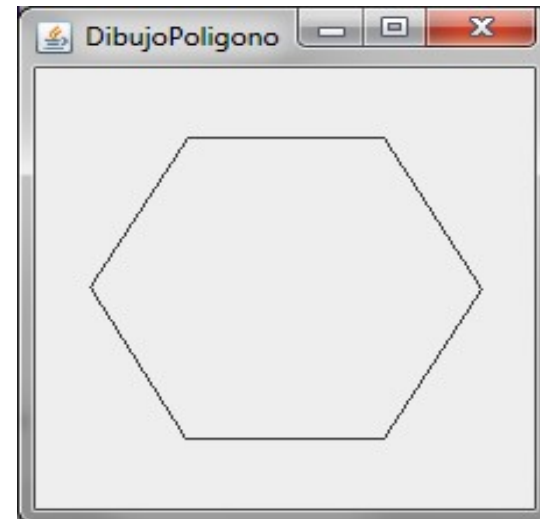
Polilínea



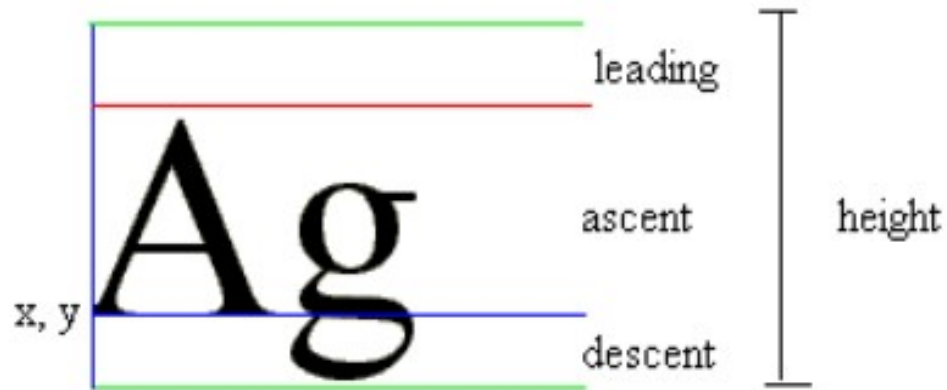
java.awt.Polygon

```
+xpoints: int[]  
+ypoints: int[]  
+npoints: int
```

```
+Polygon()  
+Polygon(xpoints: int[], ypoints: int[],  
         npoints: int)  
+addPoint(x: int, y: int)
```



Centrar un String con la clases FontMetrics



Visualizar imágenes

Aplicar el método de `drawImage` de un objeto `Graphics` para mostrar una imagen en un componente de la GUI

java.awt.Graphics

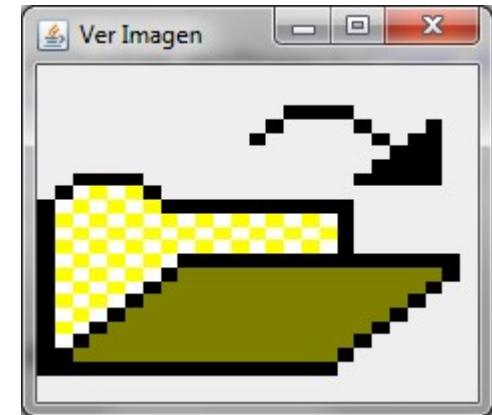
```
+drawImage(image: Image, x: int, y: int,  
    bgcolor: Color, observer:  
    ImageObserver): void
```

→

```
+drawImage(image: Image, x: int, y: int,  
    observer: ImageObserver): void
```

```
+drawImage(image: Image, x: int, y: int,  
    width: int, height: int, observer:  
    ImageObserver): void
```

```
+drawImage(image: Image, x: int, y: int,  
    width: int, height: int, bgcolor: Color,  
    observer: ImageObserver): void
```



Interfaces gráficas de usuario.

Programación dirigida por eventos

Java GUI's (Graphical User Interfaces)

Java JFC/Swing API

La clase Graphics

→ Contenedores

Layout Managers (Administradores de Presentación)

Componentes básicos

Contenedores

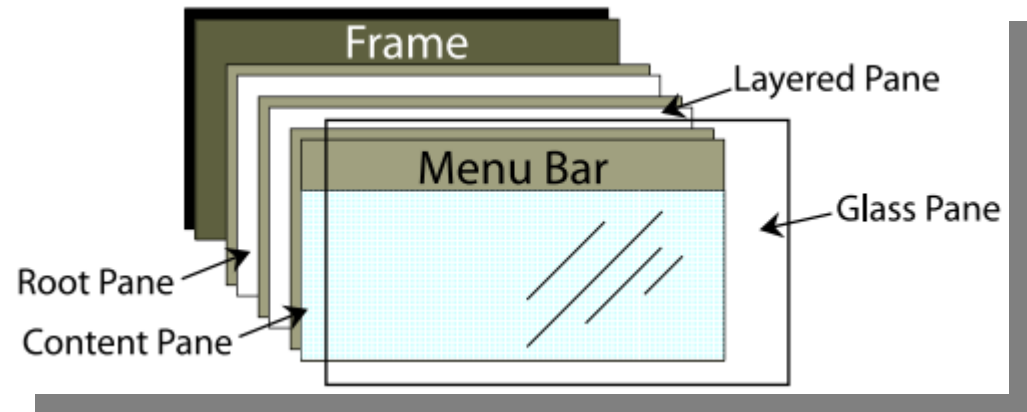
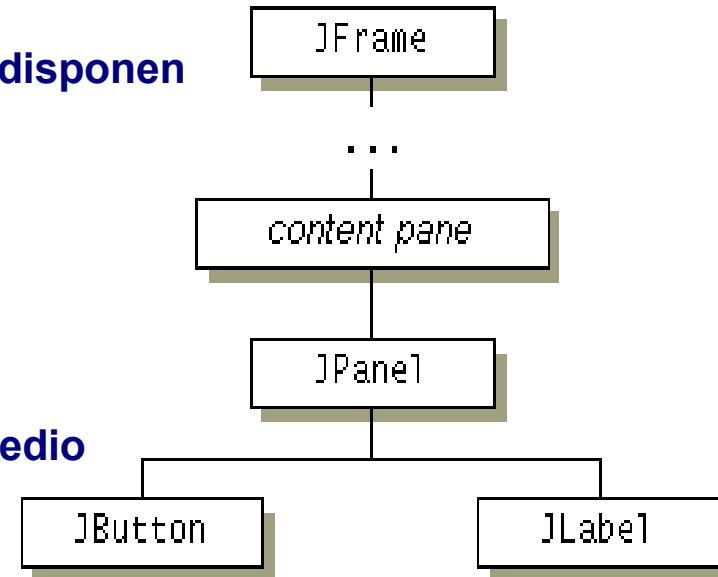
Anidamiento de componentes (Jerarquía de contenedores). Cada programa Swing contiene al menos una.

Usan un Layout Manager para determinar cómo se disponen los componentes en los contenedores

Swing provee 4 contenedores de alto nivel: JFrame, JApplet, JDialog y JWindow

La jerarquía está compuesta de diferentes capas. Cada contenedor de alto nivel contiene un contenedor intermedio conocido como “content pane”

En casi todos los programas no es necesario conocer qué hay entre el contenedor de alto nivel y el content pane



JFrame

JFrame es el contenedor superior para guardar los componentes de la GUI

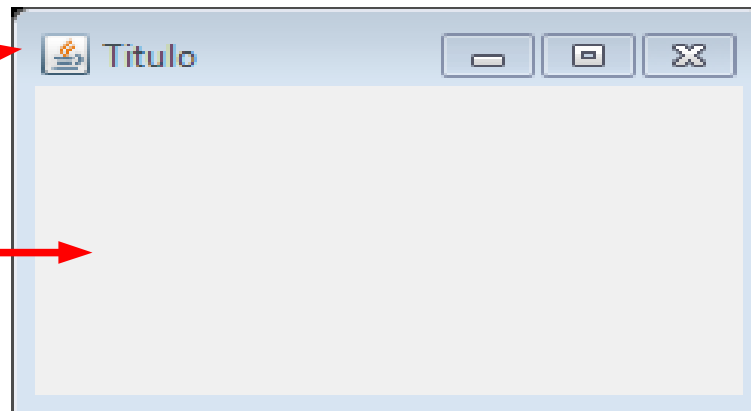
`javax.swing.JFrame`

```
+JFrame()  
+JFrame(title: String)  
+setSize(width: int, height: int): void  
+setLocation(x: int, y: int): void  
+setVisible(visible: boolean): void  
+setDefaultCloseOperation(mode: int): void  
+setLocationRelativeTo(c: Component):  
    void  
+pack(): void
```

Properties	
Class	javax.swing.JFrame
bindings	[]
alwaysOnTop	<input type="checkbox"/> false
background	<input type="checkbox"/> 240,240,240
defaultCloseOperat...	EXIT_ON_CLOSE
enabled	<input checked="" type="checkbox"/> true
font	
foreground	
iconImage	
modalExclusionType	NO_EXCLUDE
resizable	<input checked="" type="checkbox"/> true
tab order	
title	Titulo

Title bar

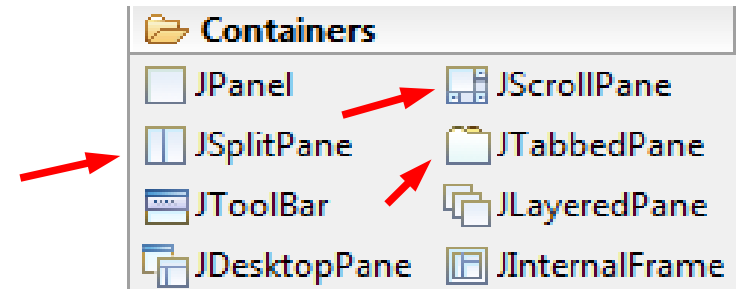
Content pane



Otros Contenedores

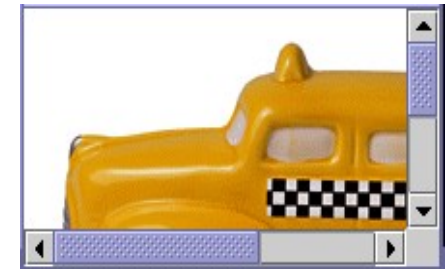
JScrollPane

Una barra de desplazamiento para desplazar el contenido de un objeto que no se ajusten completamente en el área de visualización.



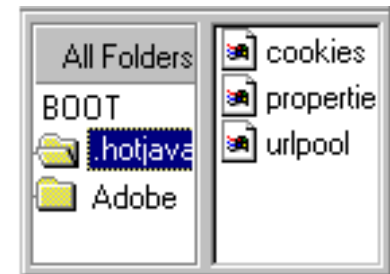
JTabbedPane

Un útil contenedor Swing que proporciona un conjunto de solapas mutuamente excluyentes para el acceso a múltiples componentes.



JSplitPane

Un contenedor Swing que contiene dos componentes con una barra separadora conocida como un divisor



Interfaces gráficas de usuario.

Programación dirigida por eventos

Java GUI's (Graphical User Interfaces)

Java JFC/Swing API

La clase Graphics

Contenedores

→ Layout Managers (Administradores de Presentación)

Componentes básicos

Layout Managers (Administradores de Presentación)


Los layout managers de Java proporcionan un nivel de abstracción que de forma automática organiza la interfaz de usuario en todos los sistemas de ventanas


En otros sistemas de ventanas los componentes gráficos son alineados utilizando medidas absolutas que funcionan bien en él pero no en otros sistemas de ventanas

El Layout Manager intenta ajustar la disposición de los componentes en el contenedor cuando se añade un nuevo componente o cuando el contenedor cambia de tamaño.

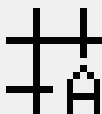
Cada contenedor tiene un Layout Manager por defecto, aunque se puede establecer otro LM para el mismo, de forma explícita. Para anular el LM por defecto se usa el método `setLayout` (para contenedores de alto nivel se usa `getContentPane().setLayout()`).

Layouts


 Absolute lay...

 FlowLayout

 BorderLayout


 GridLayout

 GridBagLayout

 CardLayout

 BoxLayout

 SpringLayout

 FormLayout

 MigLayout

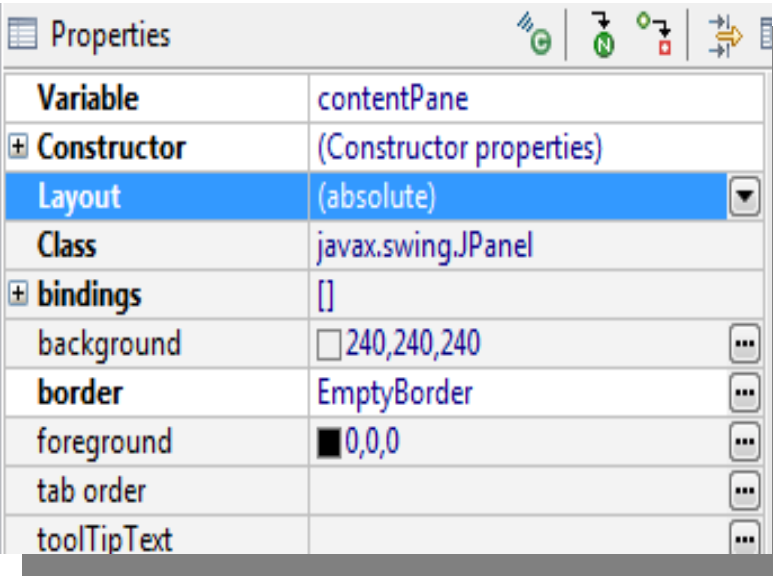
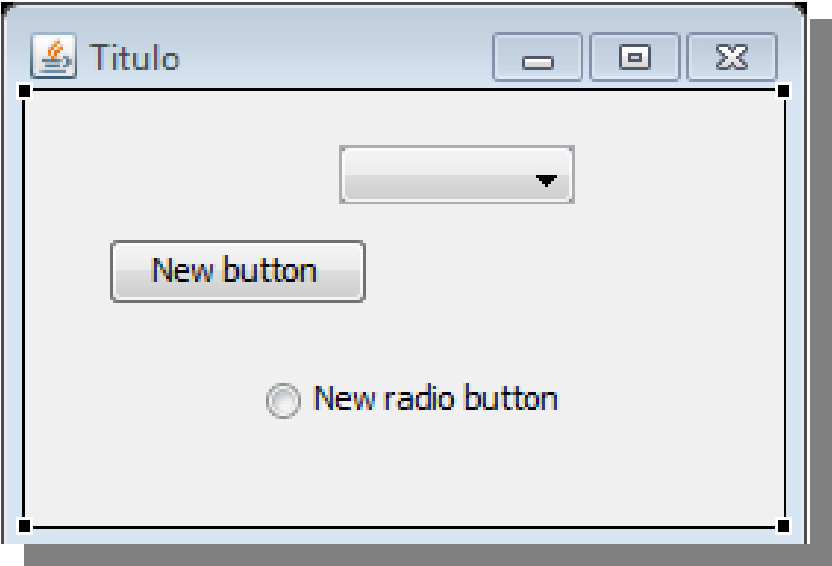
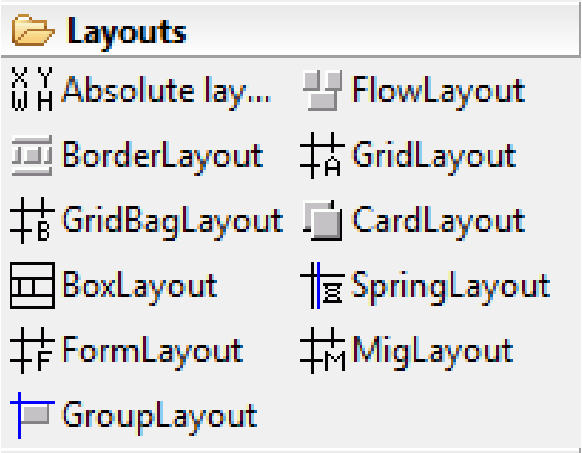
 GroupLayout

Null Layout

Java también es compatible con una disposición absoluta, llamado **null layout**, lo que le permite colocar los componentes en ubicaciones fijas.

```
setLayout(null);
```

El programador es responsable de establecer el tamaño y la posición de cada componente (setBounds(x,y,w,h))



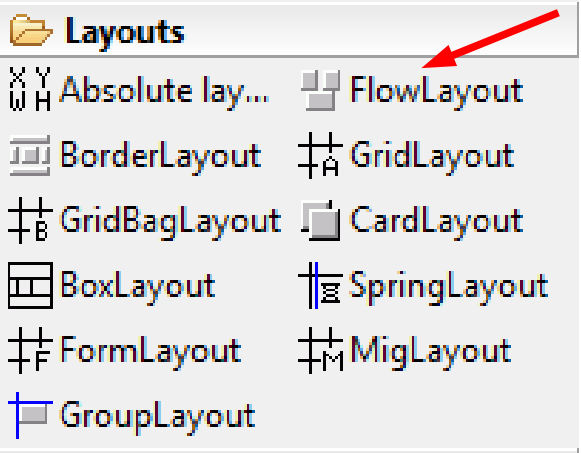
FlowLayout

Los componentes están dispuestos en el contenedor de izquierda a derecha en el orden en que se añadieron

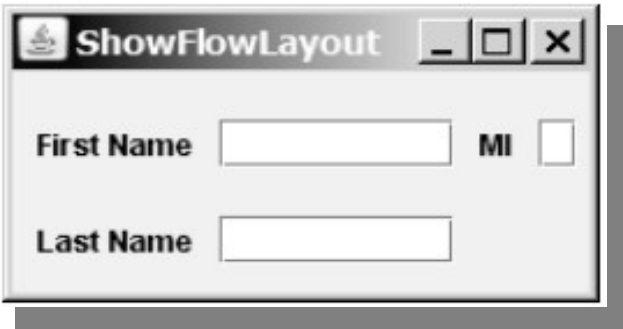
java.awt.FlowLayout

-alignment: int
-hgap: int
-vgap: int

+FlowLayout()
+FlowLayout(alignment: int)
+FlowLayout(alignment: int, hgap: int, vgap: int)



Layout	(java.awt.FlowLayout)
Class	java.awt.FlowLayout
Constructor	(Constructor properties)
alignOnBaseline	<input type="checkbox"/> false
alignment	CENTER
hgap	5
vgap	5



GridLayout

GridLayout ordena los componentes en una rejilla (matriz). Los componentes se colocan en la cuadrícula de izquierda a derecha.












En **FlowLayout** y **GridLayout**, el orden en que los componentes se añaden determina la ubicación de los componentes en el contenedor.

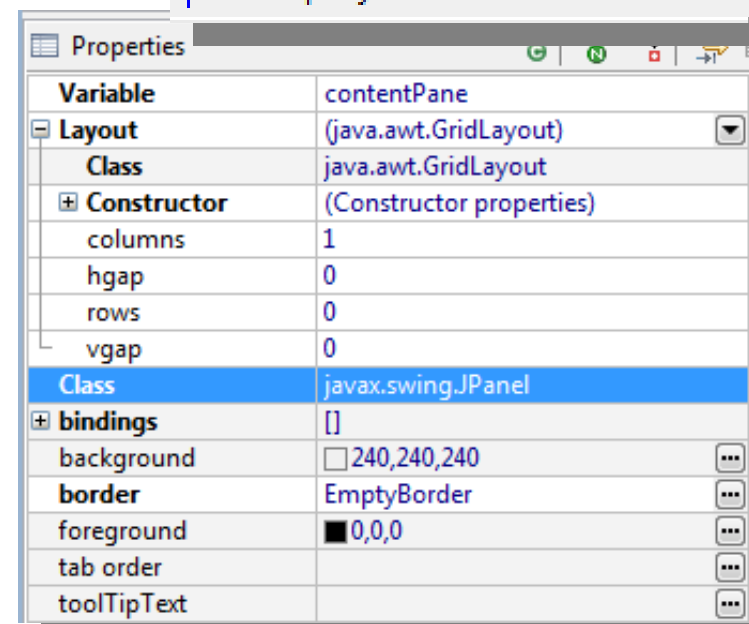
java.awt.GridLayout

```
-rows: int  
-columns: int  
-hgap: int  
-vgap: int
```

```
+GridLayout()  
+GridLayout(rows: int, columns: int)  
+GridLayout(rows: int, columns: int,  
    hgap: int, vgap: int)
```

Layouts

 Absolute lay...	 FlowLayout
 BorderLayout	 GridLayout
 GridBagLayout	 CardLayout
 BoxLayout	 SpringLayout
 FormLayout	 MigLayout
 GroupLayout	



Variable	contentPane
Layout	(java.awt.GridLayout)
Class	java.awt.GridLayout
Constructor	(Constructor properties)
columns	1
hgap	0
rows	0
vgap	0
Class	javax.swing.JPanel
bindings	[]
background	240,240,240
border	EmptyBorder
foreground	0,0,0
tab order	
toolTipText	



First Name

MI

Last Name

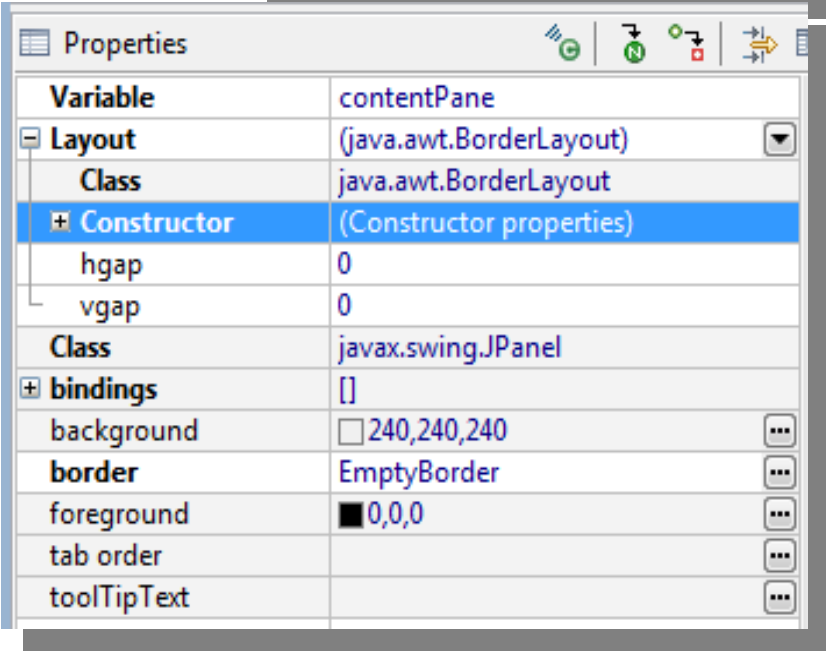
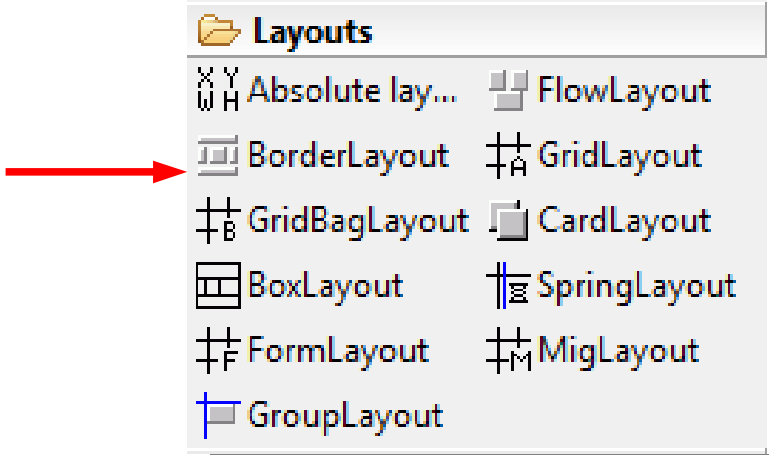
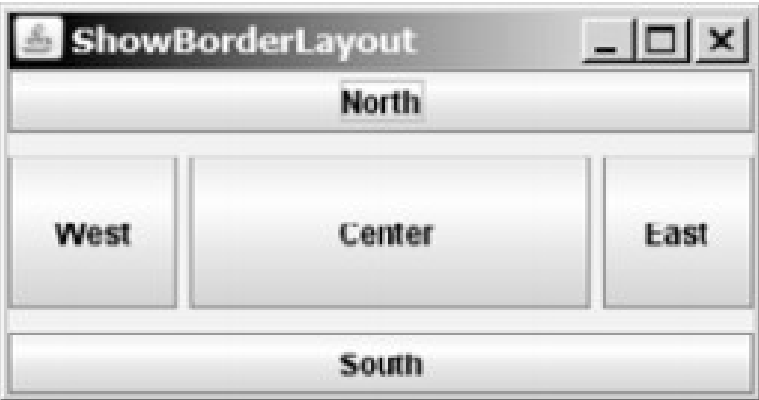
BorderLayout

BorderLayout divide un contenedor en cinco áreas: East, South, West, North,y Center

```
java.awt.BorderLayout

-hgap: int
-vgap: int

+BorderLayout()
+BorderLayout(hgap: int, vgap: int)
```



BoxLayout

BoxLayout ordena los componentes en una fila o una columna.

Todas las filas tienen el mismo alto y todas las columnas tienen el mismo ancho

Layouts

X Y W H

Absolute lay...

BorderLayout

GridBagLayout

BoxLayout

FormLayout

GroupLayout

FlowLayout

GridLayout

CardLayout

SpringLayout

MigLayout

Name Chooser

Baby names ending in O:

Arlo

Cosmo

Elmo

Hugo

Jethro

Cancel

Set

Properties

Variable	panel
Layout	(javax.swing.BoxLayout)
Class	javax.swing.BoxLayout
Constructor	(Constructor properties)
axis	X_AXIS
Class	javax.swing.JPanel
bindings	[]
background	240,240,240
border	
foreground	0,0,0
tab order	
toolTipText	

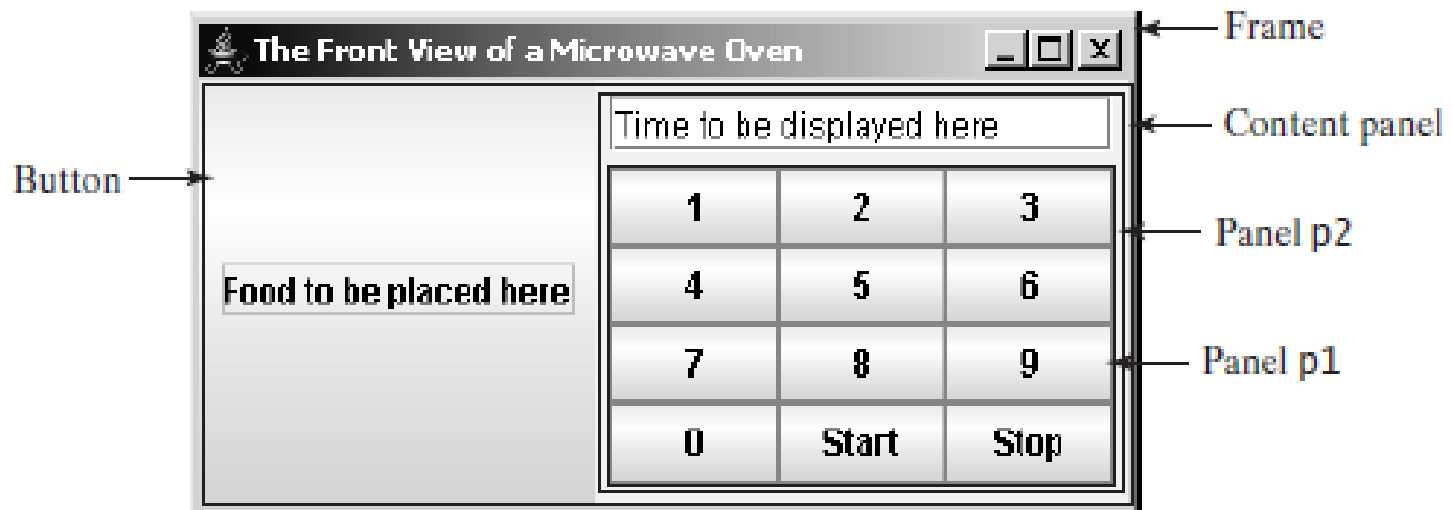
Paneles como subcontenedores

Puede dividir una ventana en paneles. Los paneles actúan como subcontenedores que agrupan los componentes de la interfaz gráfica de usuario.

La version Swing de un panel es JPanel

`new JPanel()` crea un panel con **FlowLayout** por defecto

`new JPanel(LayoutManager)` crea un panel con un layout específico



Interfaces gráficas de usuario.

Programación dirigida por eventos

Java GUI's (Graphical User Interfaces)

Java JFC/Swing API

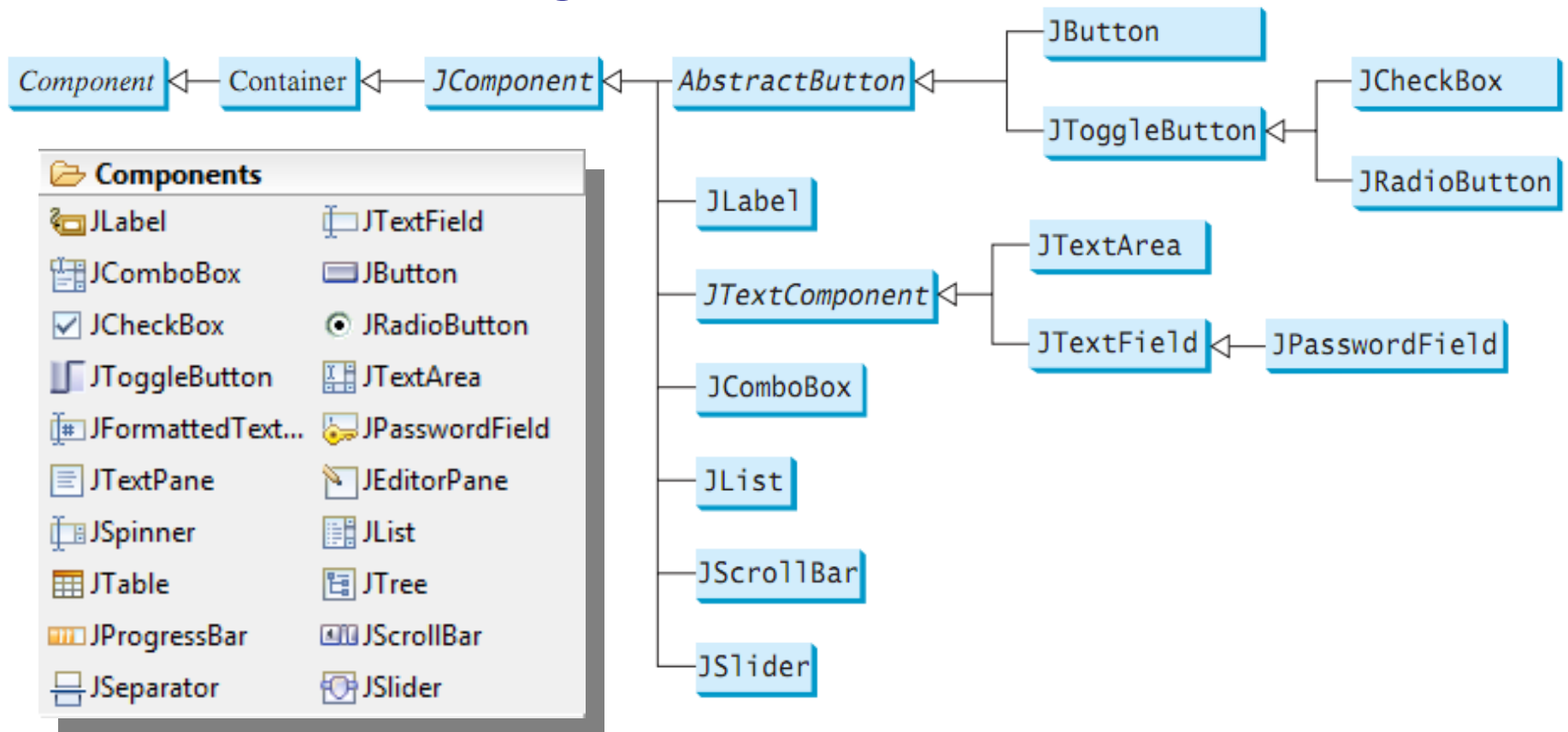
La clase Graphics

Contenedores

Layout Managers (Administradores de Presentación)

→ Componentes básicos

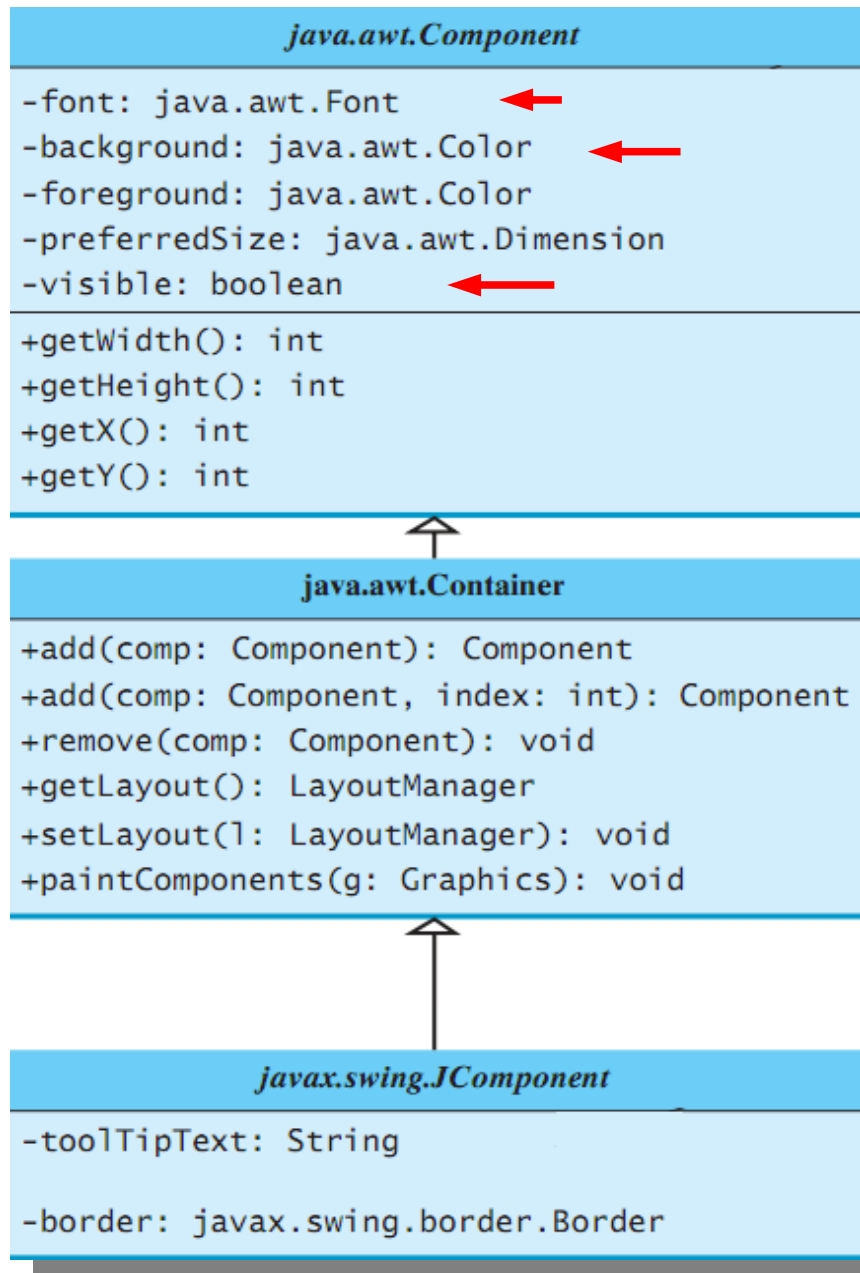
Componentes Swing GUI



Componentes más utilizados

Prefijos habituales : *jbt*, *jchk*, *jrb*, *jlbl*, *jtf*, *jpf*, *jta*, *jco*, *jlst* y *jsld*

Características comunes de los componentes GUI de Swing



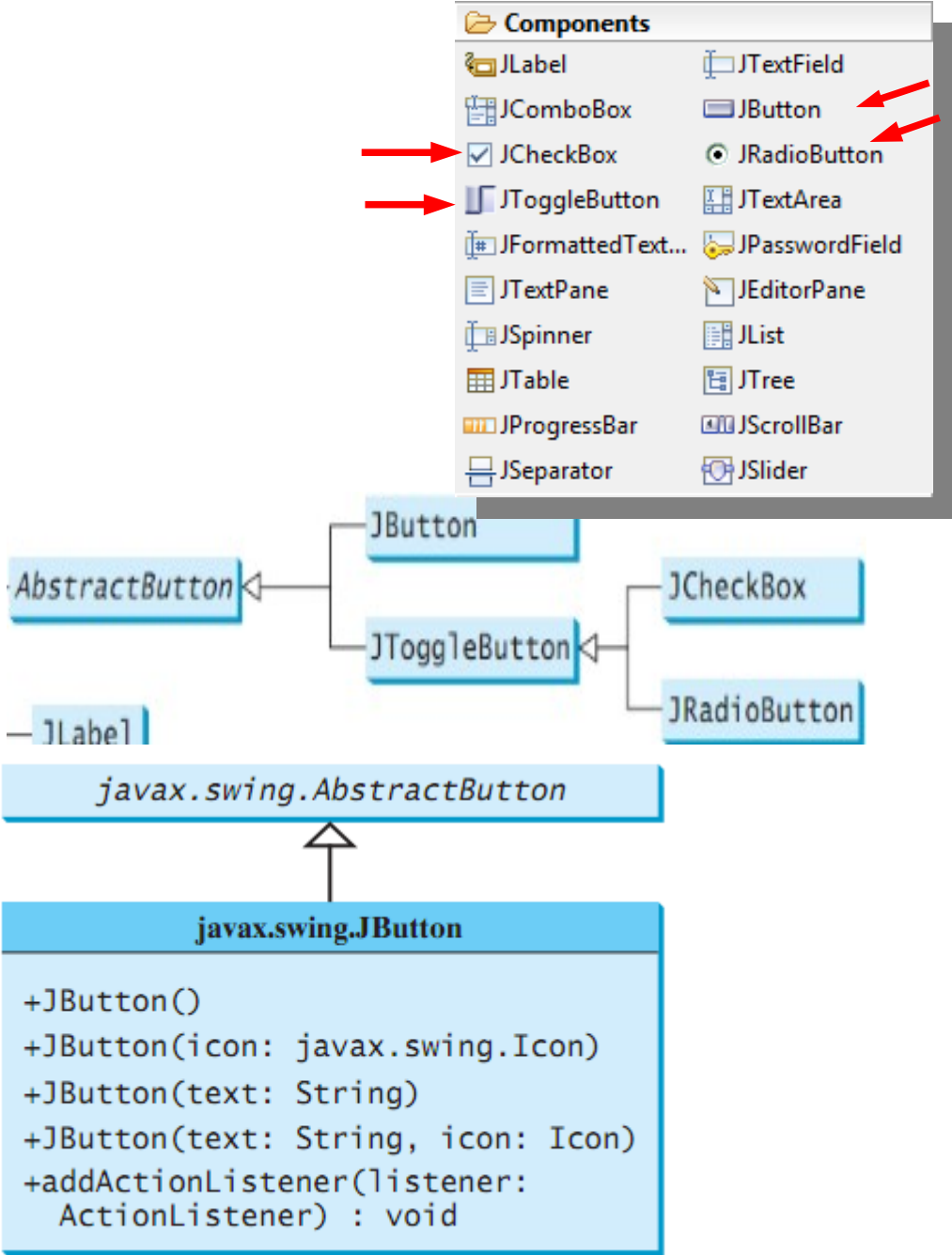
Buttons

`javax.swing.JComponent`

↑

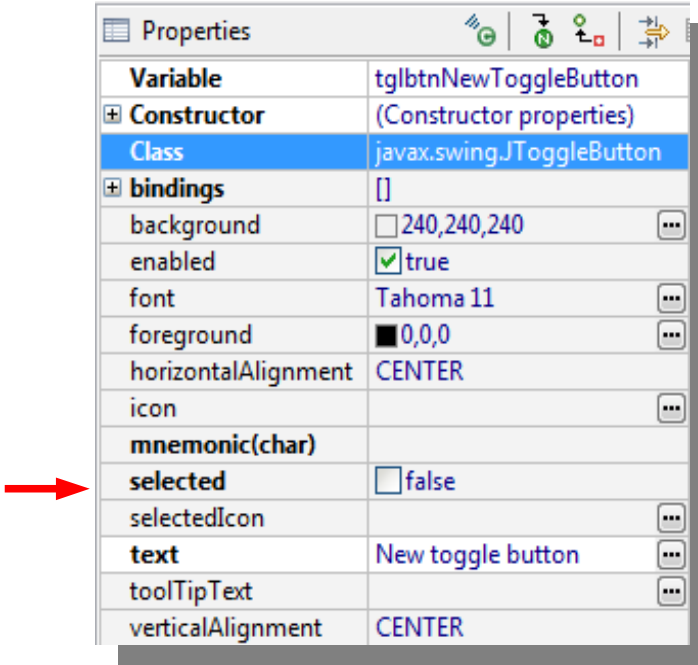
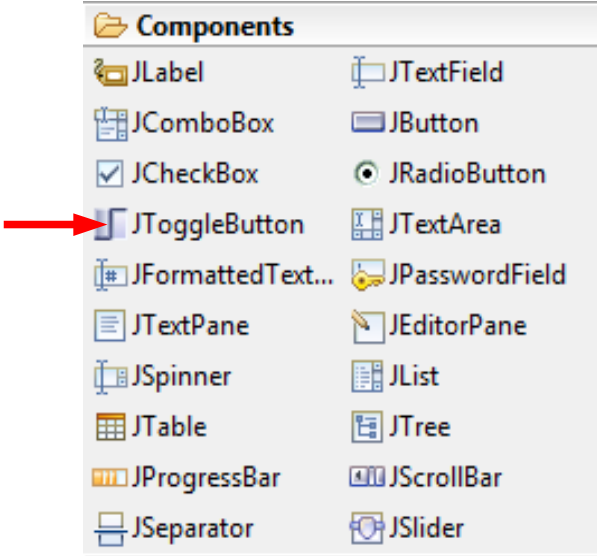
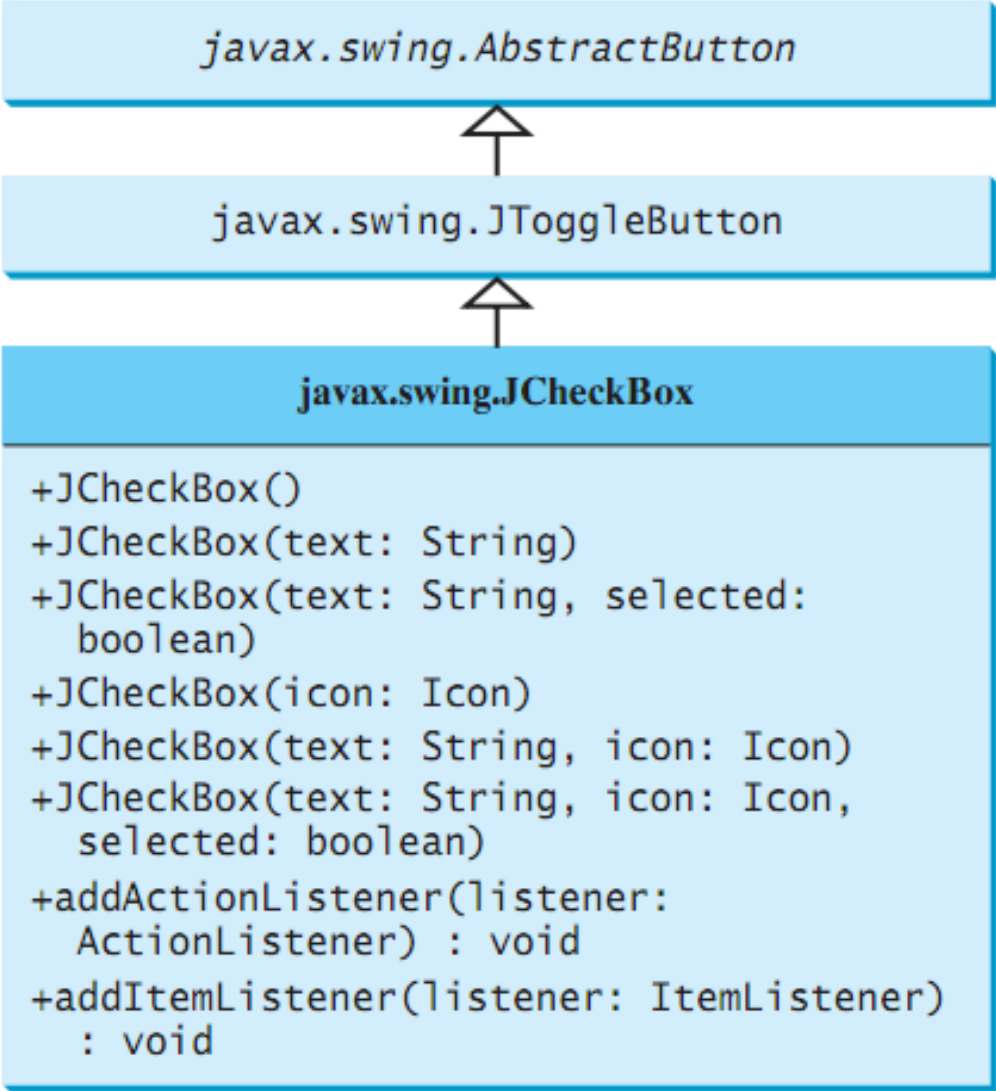
`javax.swing.AbstractButton`

- actionCommand: String
- text: String
- icon: javax.swing.Icon
- pressedIcon: javax.swing.Icon
- rolloverIcon: javax.swing.Icon
- mnemonic: int
- horizontalAlignment: int
- horizontalTextPosition: int
- verticalAlignment: int
- verticalTextPosition: int
- borderPainted: boolean
- iconTextGap: int
- selected(): boolean



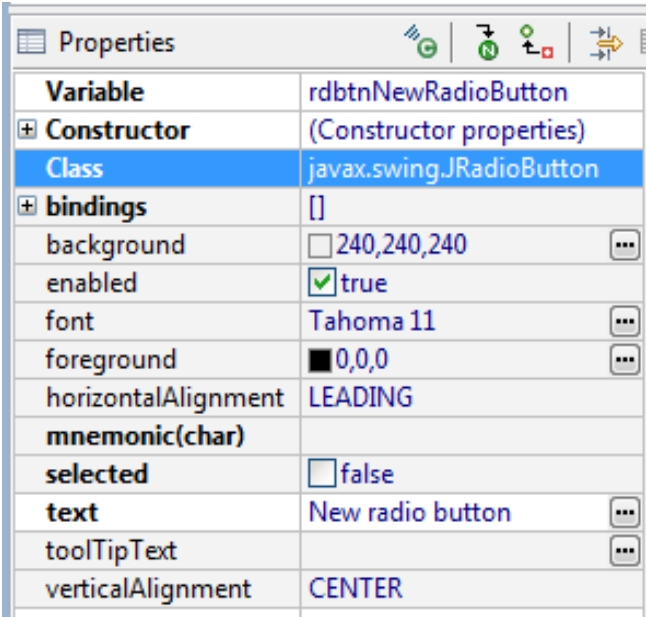
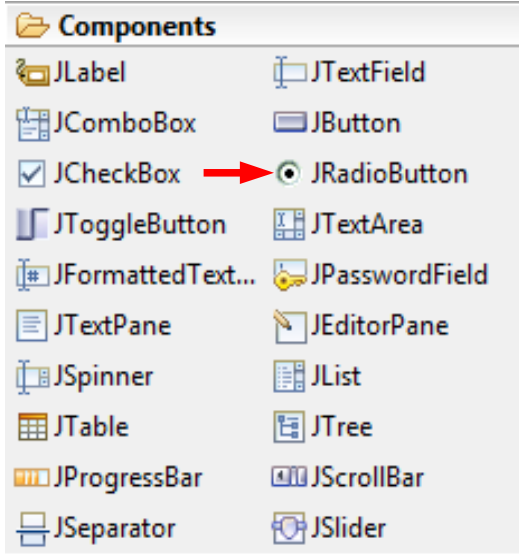
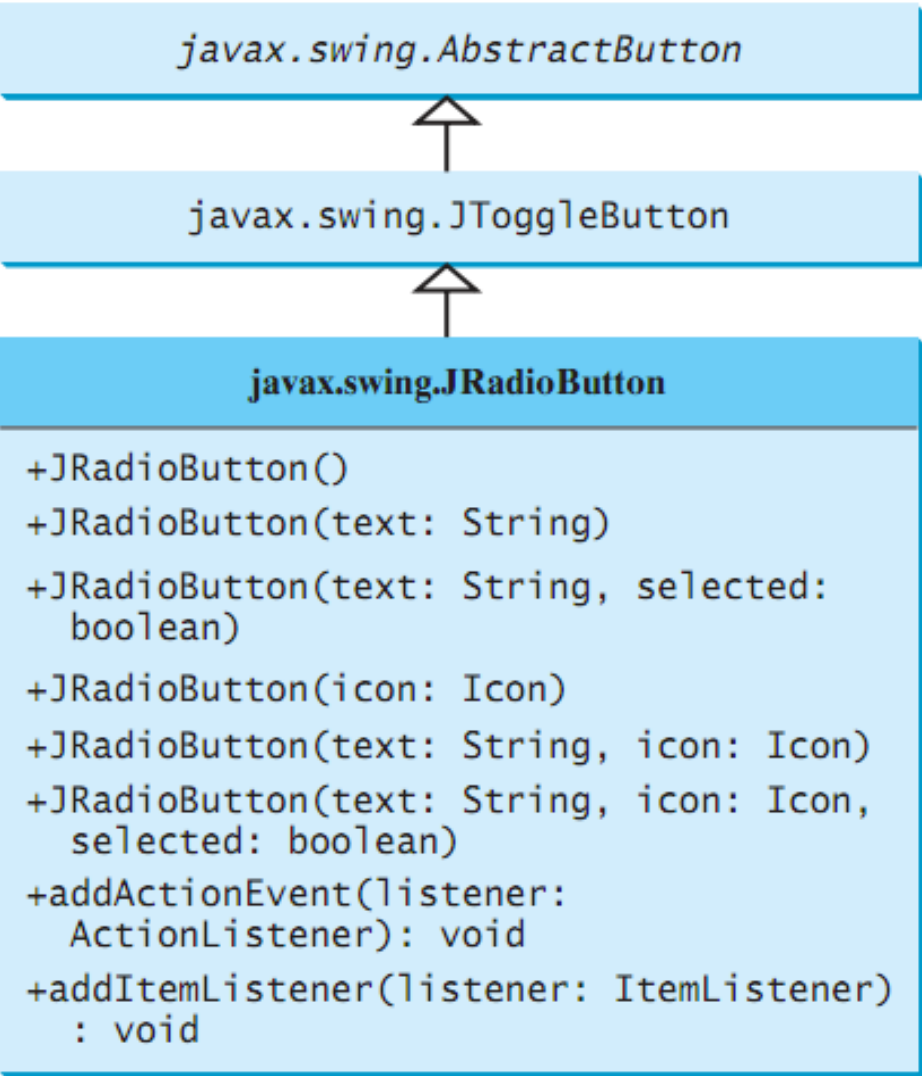
Check Boxes

JToggleButton es un botón de dos estados como un interruptor de la luz



Radio Buttons

JRadioButton, botones de opción, permiten elegir un solo elemento de un grupo de opciones



Labels

Un JLabel, etiqueta, es un área de visualización de un texto breve , una imagen, o ambas cosas

`javax.swing.JComponent`



`javax.swing.JLabel`

- text: String
 - icon: javax.swing.Icon
 - horizontalAlignment: int
 - horizontalTextPosition: int
 - verticalAlignment: int
 - verticalTextPosition: int
 - iconTextGap: int
-
- +JLabel()
 - +JLabel(icon: javax.swing.Icon)
 - +JLabel(icon: Icon, hAlignment: int)
 - +JLabel(text: String)
 - +JLabel(text: String, icon: Icon, hAlignment: int)
 - +JLabel(text: String, hAlignment: int)

Components

JLabel

JComboBox

☒ JCheckBox

JToggleButton

JFormattedText...

JTextPane

JSpinner

JTable

JProgressBar

JSeparator

JTextField

JButton

☒ JRadioButton

JTextArea

JPasswordField

JEditorPane

JList

JTree

JScrollBar

JSlider

Properties

Variable	lblNewLabel
Constructor	(Constructor properties)
Constraints	Center
Class	javax.swing.JLabel
bindings	[]
background	<input type="checkbox"/> 240,240,240
displayedMne...	
enabled	<input checked="" type="checkbox"/> true
font	Tahoma 11
foreground	<input checked="" type="checkbox"/> 0,0,0
horizontalAlign...	LEADING
icon	
labelFor	
text	New label
toolTipText	
verticalAlignment	CENTER

Text Fields

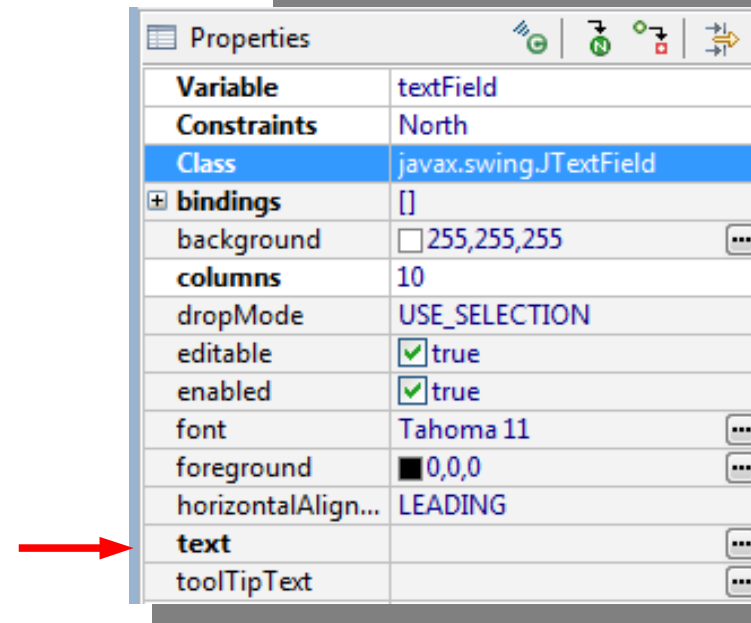
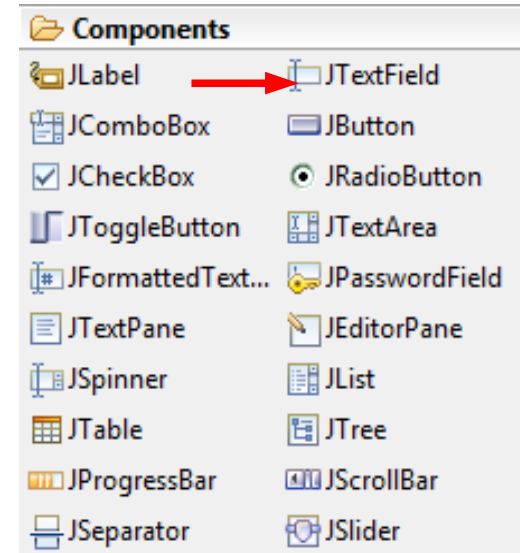
Un JTextField, campo de texto, puede ser utilizado para introducir o mostrar una cadena.

`javax.swing.text.JTextComponent`

-text: String
-editable: boolean

`javax.swing.JTextField`

-columns: int
-horizontalAlignment: int
+JTextField()
+JTextField(column: int)
+JTextField(text: String)
+JTextField(text: String, columns: int)
+addActionEvent(listener: ActionListener): void



Text Areas

Permite al usuario escribir varias líneas de texto

`javax.swing.text.JTextComponent`



`javax.swing.JTextArea`

```
-columns: int
-rows: int
-tabSize: int
-lineWrap: boolean

-wrapStyleWord: boolean

+JTextArea()
+JTextArea(rows: int, columns: int)
+JTextArea(text: String)
+JTextArea(text: String, rows: int, columns: int)
+append(s: String): void
+insert(s: String, pos: int): void
+replaceRange(s: String, start: int, end: int): void
+getLineCount(): int
```

Components

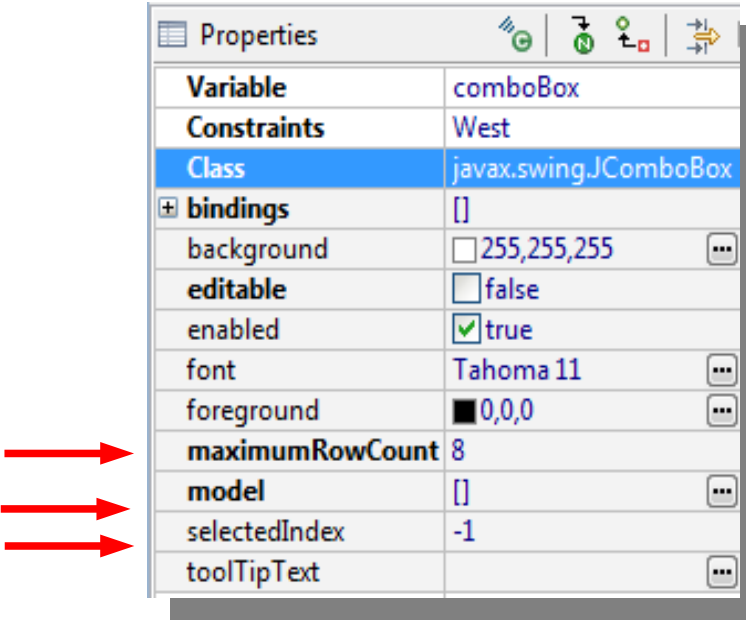
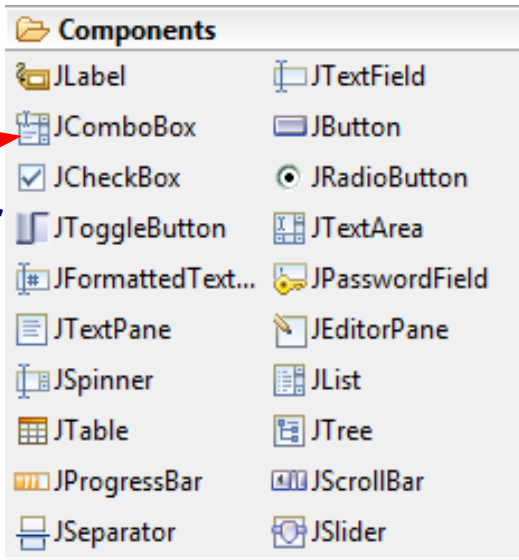
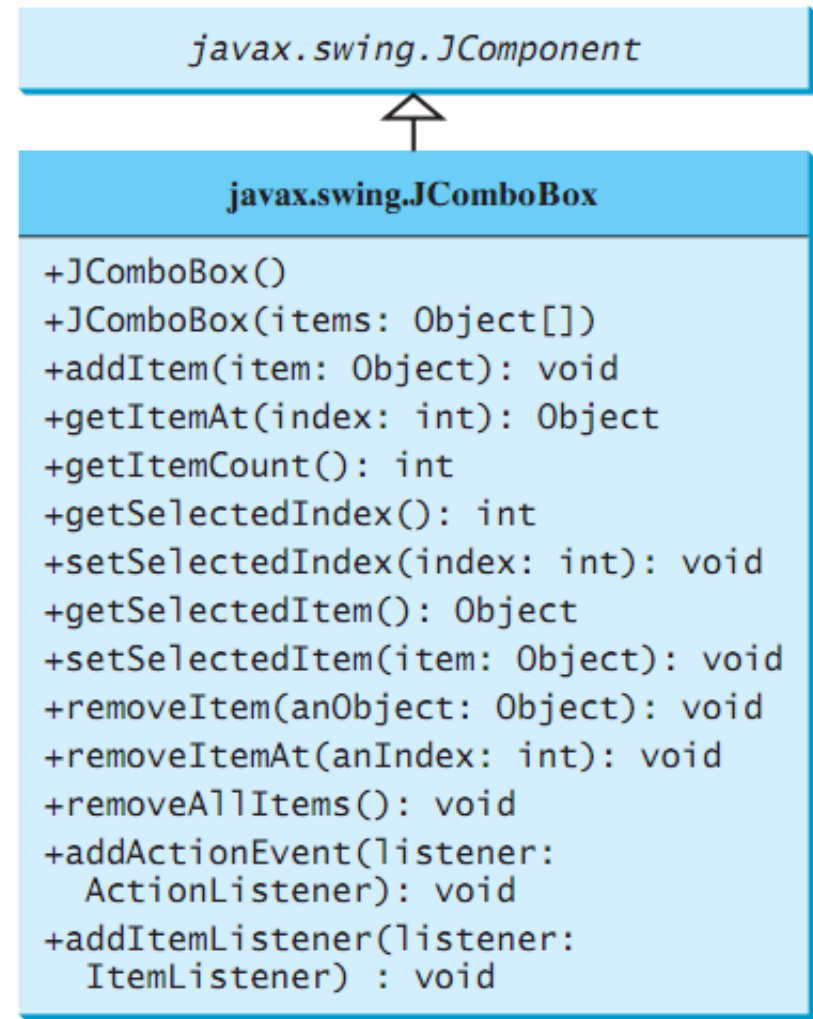
JLabel	JTextField
JComboBox	JButton
<input checked="" type="checkbox"/> JCheckBox	<input checked="" type="radio"/> JRadioButton
JToggleButton	JTextArea
JFormattedText...	JPasswordField
JTextPane	JEditorPane
JSpinner	JList
JTable	JTree
JProgressBar	JScrollBar
JSeparator	JSlider

Properties

Variable	textArea
Constraints	South
Class	javax.swing.JTextArea
bindings	[]
background	255,255,255
columns	0
dropMode	USE_SELECTION
editable	<input checked="" type="checkbox"/> true
enabled	<input checked="" type="checkbox"/> true
font	Monospaced 13
foreground	0,0,0
lineWrap	<input type="checkbox"/> false
rows	3
tabSize	8
text	
toolTipText	
wrapStyleWord	<input type="checkbox"/> false

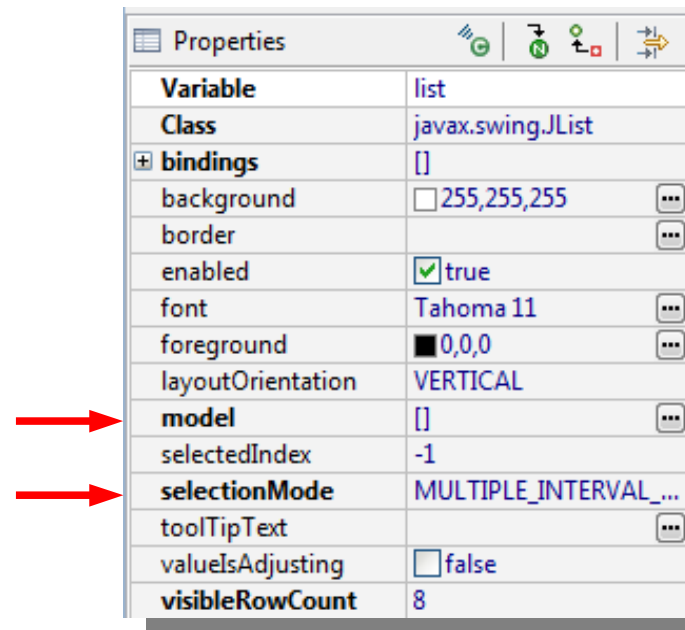
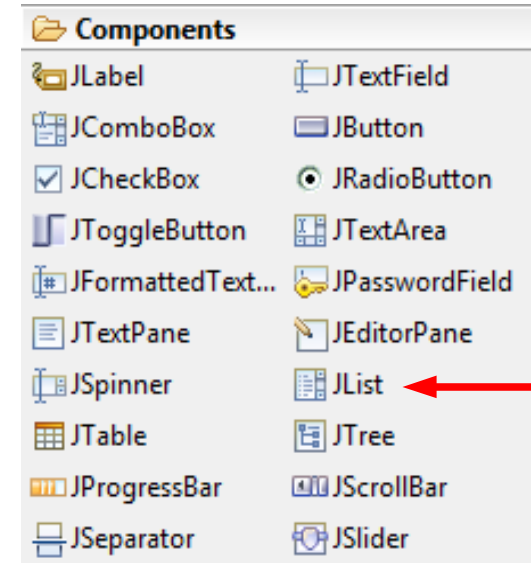
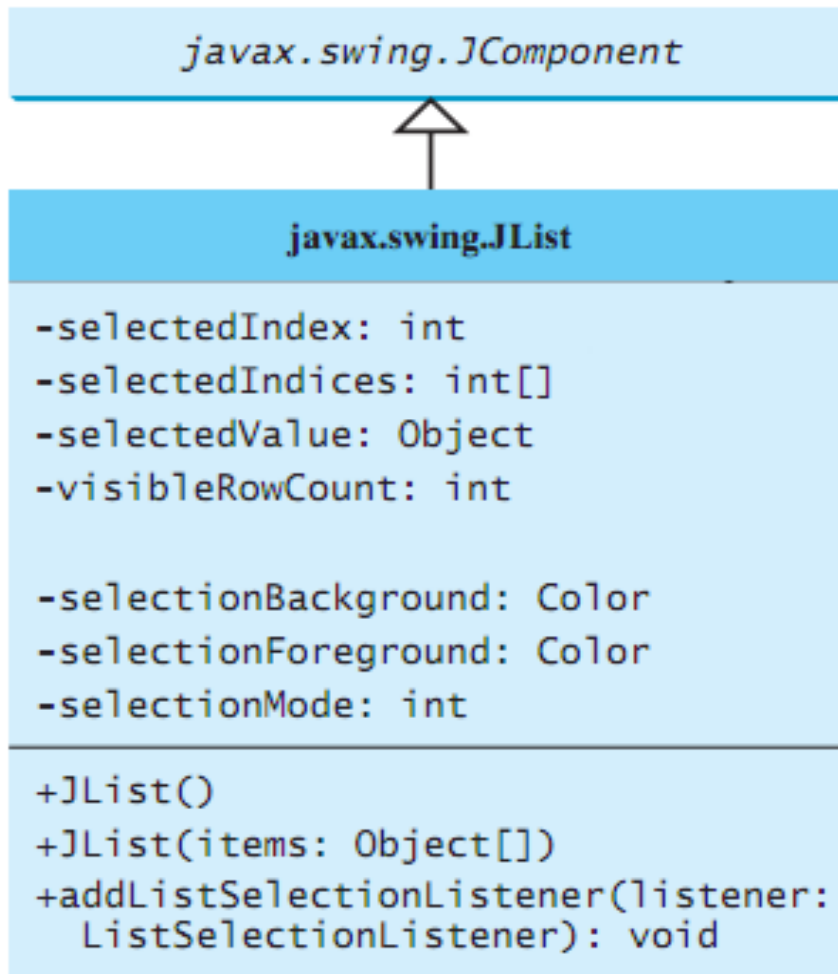
ComboBox

Un ComboBox, o una lista desplegable, contiene una lista de elementos entre cuales puede elegir el usuario. Es útil para limitar la gama de opciones al usuario y evitar la validación de entrada de datos.



Lists

UnJList es un componente que básicamente realiza la misma función como un ComboBox pero permite al usuario elegir un valor único o varios valores



Scroll Bars

JScrollBar es un componente que permite al usuario seleccionar entre una gama de valores

javafx.swing.JComponent

javafx.swing.JScrollBar

-orientation: int

-maximum: int

-minimum: int

-visibleAmount: int

-value: int

-blockIncrement: int

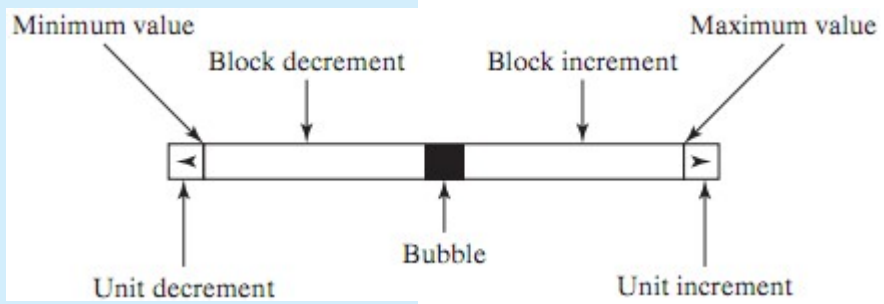
-unitIncrement: int

+JScrollBar()

+JScrollBar(orientation: int)

+JScrollBar(orientation: int, value: int, extent: int, min: int, max: int)

+addAdjustmentListener(listener: AdjustmentListener): void



Components

JLabel	JTextField
JComboBox	JButton
JCheckBox	JRadioButton
JToggleButton	JTextArea
JFormattedText...	JPasswordField
JTextPane	JEditorPane
JSpinner	JList
JTable	JTree
JProgressBar	JScrollBar
JSeparator	JSlider

Properties

Variable	scrollBar
Class	javafx.swing.JScrollBar
bindings	[]
background	200,200,200
blockIncrement	10
enabled	true
foreground	240,240,240
maximum	100
minimum	0
orientation	VERTICAL
toolTipText	
unitIncrement	1
value	0

Slider

JSlider permite al usuario seleccionar gráficamente un valor deslizando un botón dentro de un intervalo acotado

javax.swing.JComponent



javax.swing.JSlider

-maximum: int
-minimum: int
-value: int
-orientation: int
-paintLabels: boolean
-paintTicks: boolean
-paintTrack: boolean
-majorTickSpacing: int
-minorTickSpacing: int
-inverted: boolean

+JSlider()
+JSlider(min: int, max: int)
+JSlider(min: int, max: int, value: int)
+JSlider(orientation: int)
+JSlider(orientation: int, min: int, max: int, value: int)
+addChangeListener(listener: ChangeListener) :void

Components

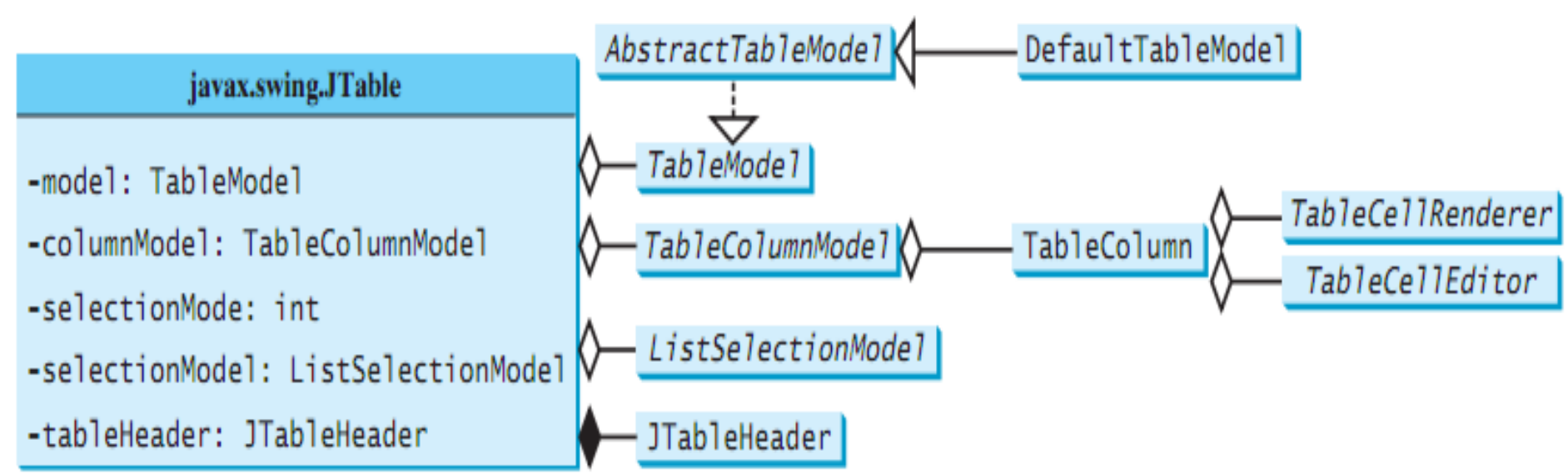
JLabel	JTextField
JComboBox	JButton
<input checked="" type="checkbox"/> JCheckBox	<input checked="" type="radio"/> JRadioButton
JToggleButton	JTextArea
<input checked="" type="checkbox"/> JFormattedText...	JPasswordField
JTextPane	JEditorPane
JSpinner	JList
JTable	JTree
JProgressBar	JScrollBar
JSeparator	JSlider



Properties

Variable	slider
Class	javax.swing.JSlider
bindings	[]
background	<input type="checkbox"/> 240,240,240
border	
enabled	<input checked="" type="checkbox"/> true
font	Tahoma 11
foreground	<input type="checkbox"/> 240,240,240
majorTickSpacing	0
maximum	100
minimum	0
minorTickSpacing	0
orientation	HORIZONTAL
paintLabels	<input type="checkbox"/> false
paintTicks	<input type="checkbox"/> false
paintTrack	<input checked="" type="checkbox"/> true
snapToTicks	<input type="checkbox"/> false
toolTipText	
value	50

JTable JTable visualiza datos en una tabla



JTable no soporta directamente el desplazamiento (Scrol). Para crear una tabla con desplazamiento, es necesario crear un JScrollPane y agregar una instancia de Jtable.

JTable tiene tres modelos: un table model, un column model y un list-selecction model

javax.swing.JTable

- autoCreateColumnsFromModel: boolean
- autoResizeMode: int
- cellEditor: TableCellEditor
- columnModel: TableColumnModel
- columnSelectionAllowed: boolean
- editingColumn: int
- editingRow: int
- gridColor: java.awt.Color
- intercellSpacing: Dimension
- model: TableModel
- rowCount: int
- rowHeight: int
- rowMargin: int
- rowSelectionAllowed: boolean
- selectionBackground: java.awt.Color
- selectionForeground: java.awt.Color
- showGrid: boolean
- selectionMode: int
- selectionModel: ListSelectionModel
- showHorizontalLines: boolean
- showVerticalLines: boolean
- tableHeader: JTableHeader

La clase JTable contiene siete constructores

- +JTable()
- +JTable(numRows: int, numColumns: int)
- +JTable(rowData: Object[][], columnData: Object[])
- +JTable(dm: TableModel)
- +JTable(dm: TableModel, cm: TableColumnModel)
- +JTable(dm: TableModel, cm: TableColumnModel, sm: ListSelectionModel)
- +JTable(rowData: Vector, columnNames: Vector)
- +addColumn(aColumn: TableColumn): void
- +clearSelection(): void
- +editCellAt(row: int, column: int): void
- +getDefaultEditor(column: Class): TableCellEditor
- +getDefaultRenderer(col: Class): TableCellRenderer
- +setDefaultEditor(column: Class, editor: TableCellEditor): void
- +setDefaultRenderer(column: Class, editor: TableCellRenderer): void

JTable

Properties	
Variable	jTable1
Constructor	(Constructor properties)
Class	javax.swing.JTable
bindings	[]
background	<input type="checkbox"/> 255,255,255
border	...
cellSelectionEna...	<input type="checkbox"/> false
columnSelection...	<input type="checkbox"/> false
enabled	<input checked="" type="checkbox"/> true
fillsViewportHeig...	<input type="checkbox"/> false
font	Tahoma 11
foreground	<input checked="" type="checkbox"/> 0,0,0
model	...
rowSelectionAllo...	<input checked="" type="checkbox"/> true
selectionMode	
showGrid	<input checked="" type="checkbox"/> true
showHorizontalL...	<input checked="" type="checkbox"/> true
showVerticalLines	<input checked="" type="checkbox"/> true
surrendersFocus...	<input type="checkbox"/> false
toolTipText	...

Components	
JLabel	JTextField
JComboBox	JButton
<input checked="" type="checkbox"/> JCheckBox	<input checked="" type="radio"/> JRadioButton
JToggleButton	JTextArea
JFormattedText...	JPasswordField
JTextPane	JEditorPane
JSpinner	JList
JTable	JTree
JProgressBar	JScrollBar
JSeparator	JSlider

TableModel y TableColumnModel

«interface»
javax.swing.table.TableModel

```
+getColumnClass(columnIndex: int): Class  
+getColumnName(columnIndex: int): String  
+getColumnCount(): int  
+getRowCount(): int  
+getValueAt(rowIndex: int, columnIndex: int):  
    Object  
+setValueAt(aValue: Object, rowIndex:  
    int, columnIndex: int): void  
+isCellEditable(rowIndex: int, columnIndex:  
    int): boolean  
+addTableModelListener(l:  
    TableModelListener): void  
+removeTableModelListener(l:  
    TableModelListener): void
```

javax.swing.table.AbstractTableModel



javax.swing.table.DefaultTableModel

```
+DefaultTableModel()  
+DefaultTableModel(rowCount: int, columnCount: int)  
+DefaultTableModel(columnNames: Object[], rowCount: int)  
+DefaultTableModel(data: Object[][], columnNames: Object[])  
+DefaultTableModel(columnNames: Vector, rowCount: int)  
+DefaultTableModel(data: Vector, columnNames: Vector)  
+DefaultTableModel(rowData: Vector, columnNames: Vector)  
+addColumn(columnName: Object): void  
+addColumn(columnName: Object, columnData: Vector)  
+addRow(rowData: Object[]): void  
+addRow(rowData: Vector): void  
+getColumnCount(): int  
+getDataVector(): Vector  
+getRowCount(): int  
+insertRow(row: int, rowData: Object[]): void  
+insertRow(row: int, rowData: Vector): void  
+removeRow(row: int): void  
+setColumnCount(columnCount: int): void  
+setColumnIdentifiers(newIdentifiers: Object[]): void  
+setColumnIdentifiers(columnIdentifiers: Vector): void  
+setDataVector(dataVector: Object[][], columnIdentifiers:  
    Object[]): void  
+setDataVector(dataVector: Vector, columnIdentifiers: Vector):  
    void  
+setRowCount(rowCount: int): void
```

TableModel gestiona los datos de la tabla. Puede añadir y eliminar filas a través de un TableModel. También puede agregar una columna a través de un TableModel.

TableColumnModel

«interface»
javax.swing.table.TableColumnModel

```
+addColumn(aColumn: TableColumn): void  
+getColumn(columnIndex: int): TableColumn  
+getColumnCount(): int  
+getColumnIndex(columnIdentifier: Object): int  
+getColumnMargin(): int  
+getColumns(): Enumeration  
+getColumnSelectionAllowed(): boolean  
+getSelectedColumnCount(): int  
+getSelectedColumns(): void  
+getSelectionModel(): ListSelectionModel  
+getTotalColumnWidth(): int  
+moveColumn(columnIndex: int, newIndex: int): void  
+removeColumn(column: TableColumn): void  
+setColumnMargin(newMargin: int): void  
+setColumnSelectionAllowed(flag: boolean): void  
+setSelectionModel(newModel: ListSelectionModel): void
```

--- javax.swing.table.DefaultTableColumnModel

◆ javax.swing.table.TableColumn

TableColumnModel gestiona las columnas de una tabla y se pueden utilizar para seleccionar, añadir, mover y quitar columnas de la tabla.

`javax.swing.table.TableColumn`

```
#cellEditor: TableCellEditor  
#cellRenderer: TableCellRenderer  
#headerRenderer: TableCellRenderer  
#headerValue: Object  
#identifier: Object  
#maxWidth: int  
#minWidth: int  
#modelIndex: int  
#preferredWidth: int  
#resizable: boolean  
#width: int
```

```
+TableColumn()  
+TableColumn(modelIndex: int)  
+TableColumn(modelIndex: int, width: int)  
+TableColumn(modelIndex: int, width: int,  
    cellRenderer: TableCellRenderer)  
+setWidthToFit(): void
```

La clase `TableColumn` se utiliza para modelar una columna individual en la tabla

javax.swing.JComponent



javax.swing.table.JTableHeader

#columnModel: TableColumnModel
#draggedColumn: TableColumn
#draggedDistance: TableCellRenderer
#reorderingAllowed: boolean
#resizingAllowed: boolean
#resizingColumn: TableColumn
#table: JTable

+JTableHeader()
+JTableHeader(cm: TableColumnModel)

La clase JTableHeader muestra el encabezado de la JTable

AutoSort y Filtering

Para habilitar la ordenación en cualquier columna en un JTable debe crear una instancia de TableRowSorter con un TableModel y asignar RowSorter JTable de la siguiente manera:

```
TableRowSorter<TableModel> sorter =  
    new TableRowSorter<TableModel>(tableModel);  
jTable.setRowSorter(sorter);
```

Puede especificar un filtro para seleccionar filas de la tabla. El filtro se puede aplicar sobre una columna o todas las columnas

```
RowFilter rowFilter = RowFilter.regexFilter("U.*", int[]{0, 1});
```

```
Object[][] datos = new Object[muni.size()][cabe.size()];
```

```
int i = 0;
```

```
for (Iterator<Municipio> iterator = muni.iterator(); iterator.hasNext();) {
```

```
    Municipio municipio = iterator.next();
```

```
    datos[i][0] = municipio.getComunidad();
```

```
    datos[i][1] = municipio.getProvincia();
```

```
    datos[i][2] = municipio.getMunicipio();
```

```
    datos[i][3] = municipio.getPoblacion();
```

```
    i++;
```

```
}
```

```
Object[] cabecera = new Object[cabe.size()];
```

```
int j = 0;
```

```
for (Iterator iterator = cabe.iterator(); iterator.hasNext();) {
```

```
    String string = (String) iterator.next();
```

```
    cabecera[j] = string;
```

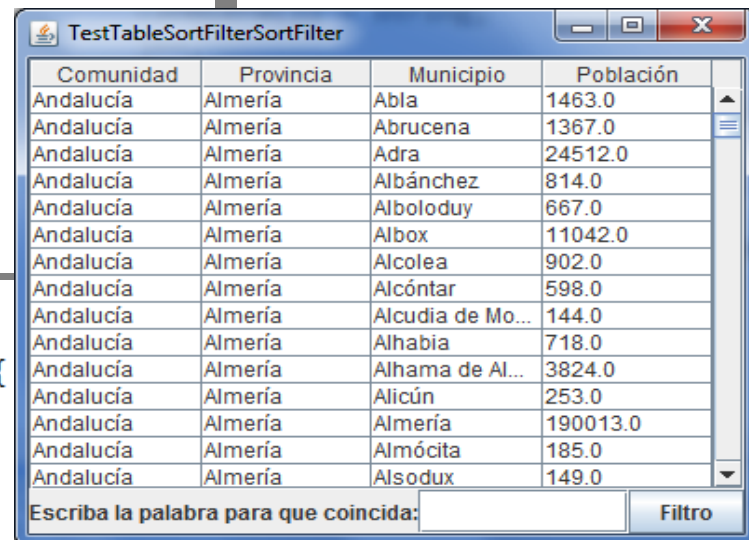
```
    j++;
```

```
}
```

```
DefaultTableModel tableModel = new DefaultTableModel(datos, cabecera);
```

```
JTable1 = new JTable(tableModel);
```

```
sorter = new TableRowSorter<TableModel>(JTable1.getModel());
```



Comunidad	Provincia	Municipio	Población
Andalucía	Almería	Abla	1463.0
Andalucía	Almería	Abrucena	1367.0
Andalucía	Almería	Adra	24512.0
Andalucía	Almería	Albánchez	814.0
Andalucía	Almería	Alboloduy	667.0
Andalucía	Almería	Albox	11042.0
Andalucía	Almería	Alcolea	902.0
Andalucía	Almería	Alcóntar	598.0
Andalucía	Almería	Alcudia de Mo...	144.0
Andalucía	Almería	Alhabia	718.0
Andalucía	Almería	Alhama de Al...	3824.0
Andalucía	Almería	Alicún	253.0
Andalucía	Almería	Almería	190013.0
Andalucía	Almería	Almócita	185.0
Andalucía	Almería	Alsodux	149.0

Escriba la palabra para que coincida:

```
btFiltro.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent e) {
```

```
        String text = jtFiltro.getText();
```

```
        if (text.trim().length() == 0)
```

```
            sorter.setRowFilter(null);
```

```
        else
```

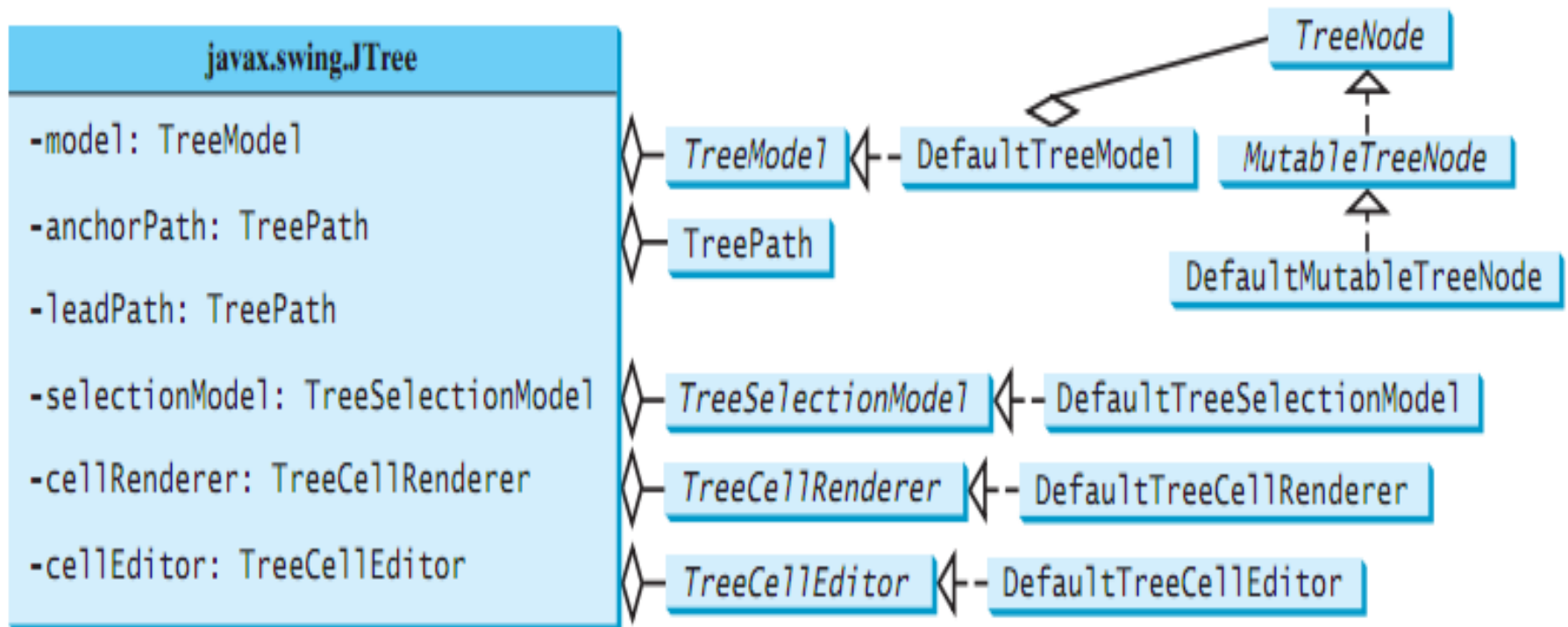
```
            sorter.setRowFilter(RowFilter.regexFilter(text));
```

```
    }
```

```
});
```

JTree

JTree es un componente Swing que muestra los datos en una jerarquía arborescente



Todos los nodos mostrados en el árbol están en la forma de una lista jerárquica indexado. El árbol puede ser utilizado para navegar por los datos estructurados, con relaciones jerárquicas. Un nodo puede tener nodos secundarios. Un nodo se llama hoja si no tiene hijos; un nodo sin padre se llama raíz de su árbol. Un árbol puede consistir en muchos subárboles de cada nodo actúa como la raíz de su propia subárbol.

JTree

Se puede crear un árbol con su no-arg constructor, un modelo de árbol, un nodo de árbol, un HashTable, un array o un vector.

javax.swing.JTree

```
#cellEditor: TreeCellEditor
#cellRenderer: TreeCellRenderer
#editable: boolean
#model: TreeModel
#rootVisible: boolean
#rowHeight: int

#scrollsOnExpand: boolean

#selectionModel: TreeSelectionModel
#showsRootHandles: boolean
#toggleClickCount: int
-anchorSelectionPath: TreePath
-expandsSelectedPaths: boolean
-leadSelectionPaths: TreePath
```

```
+JTree()
+JTree(value: java.util.Hashtable)

+JTree(value: Object[])

+JTree(newModel: TreeModel)
+JTree(root: TreeNode)
+JTree(root: TreeNode, asksAllowsChildren: boolean)
+JTree(value: Vector)

+addSelectionPath(path: TreePath): void
+addSelectionPaths(paths: TreePath[]): void
+addSelectionRow(row: int): void
+addSelectionRows(rows: int[]): void
+clearSelection(): void
+collapsePath(path: TreePath): void

+getSelectionPath(): TreePath
+getSelectionPaths(): TreePath[]
+getLastSelectedPathComponent()
+getRowCount(): int
+removeSelectionPath(path: TreePath): void
+removeSelectionPaths(paths: TreePath[]): void
```