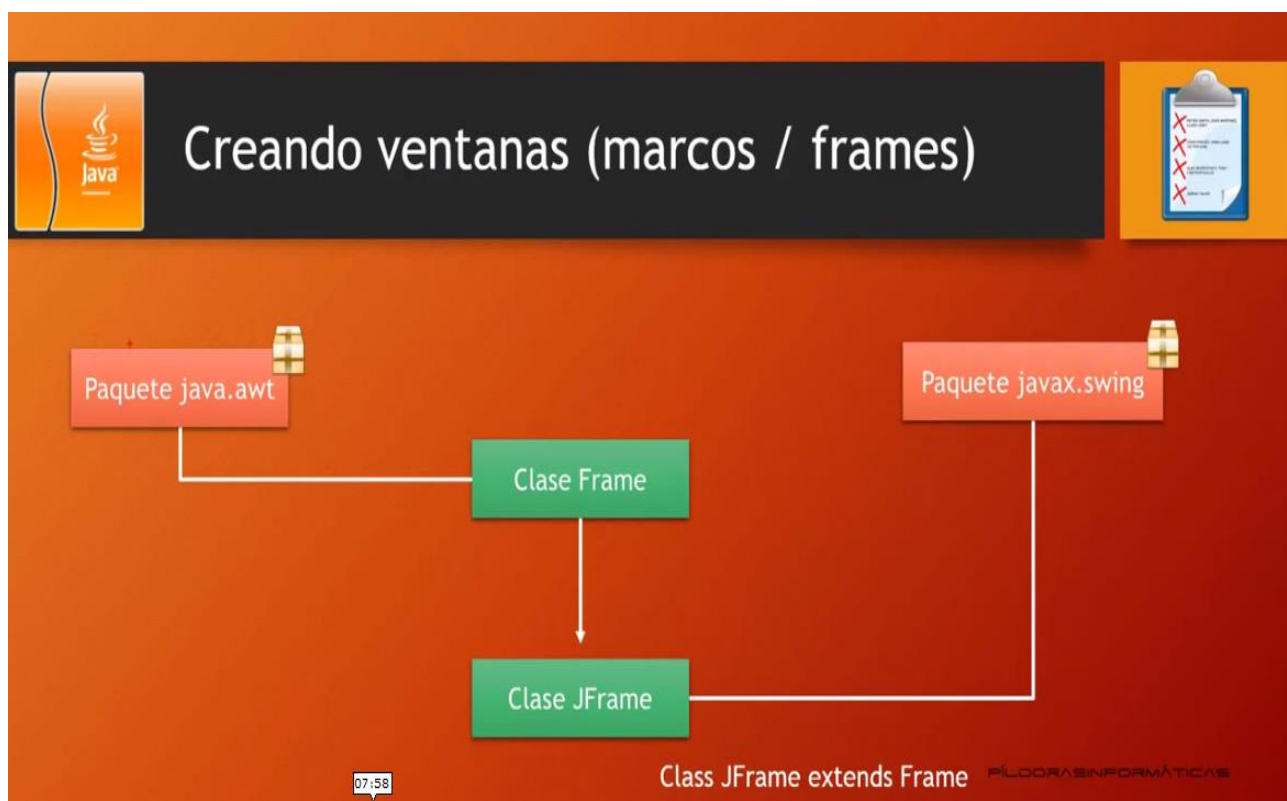
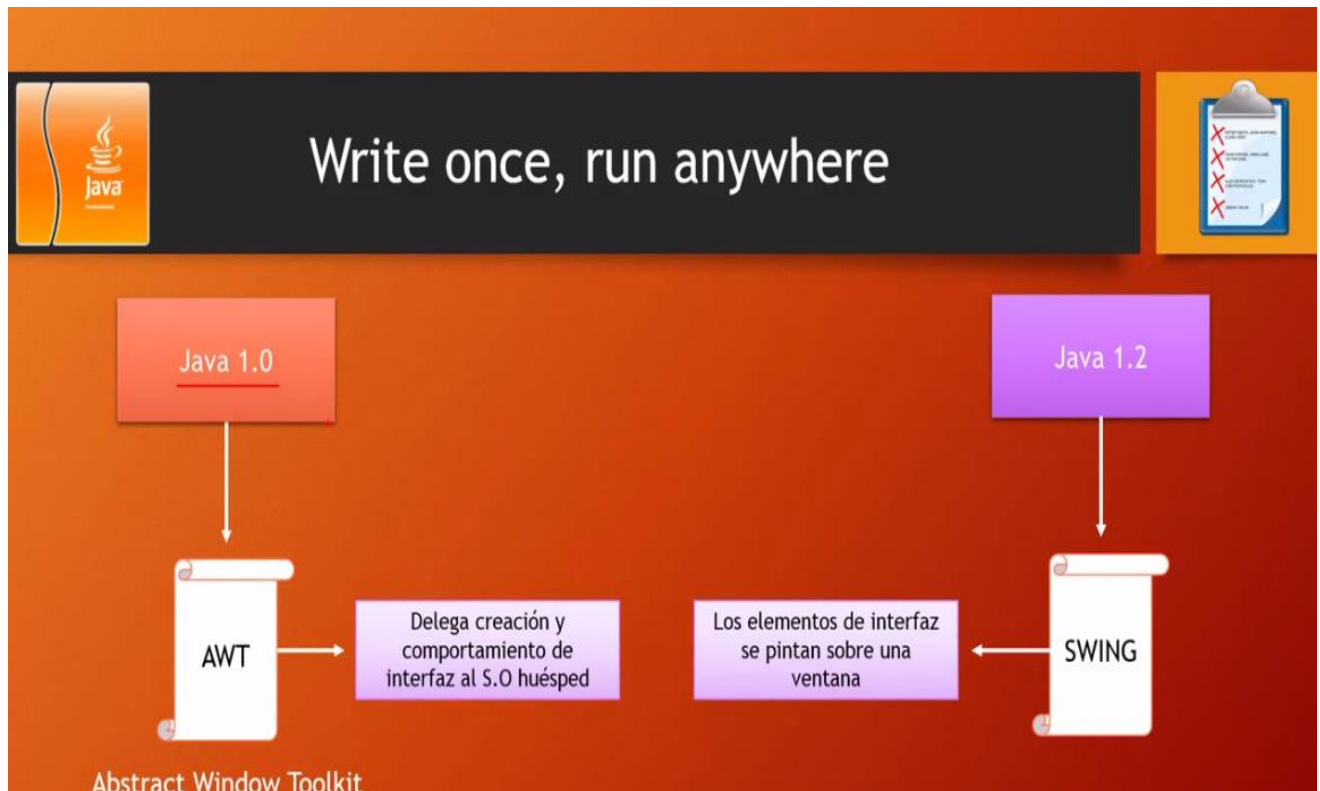


Creando un Marco

Java un lenguaje Multiplataforma.





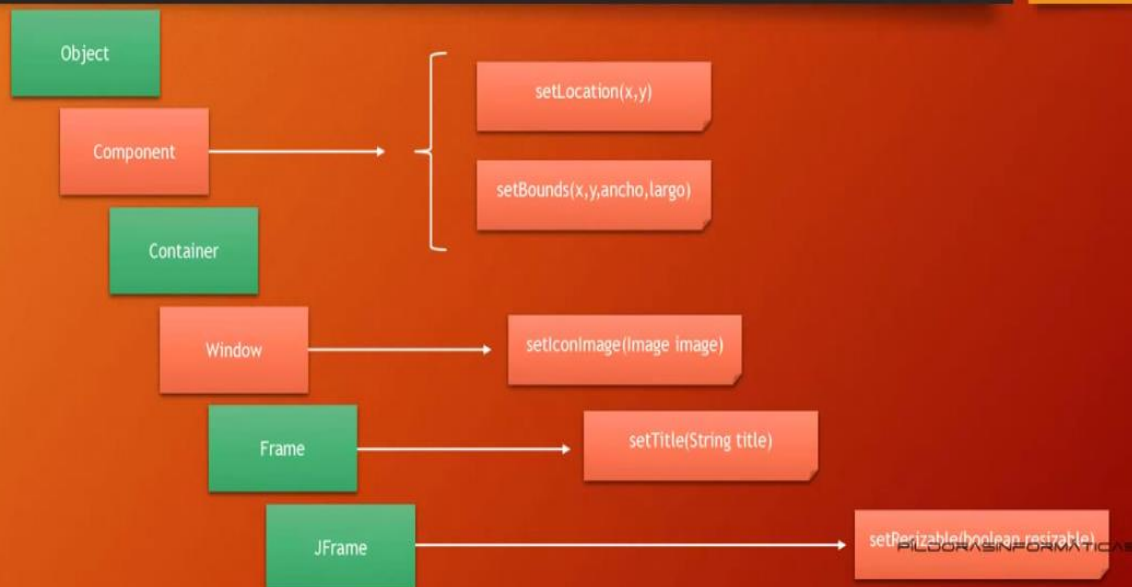
Características de los frames



- Nacen invisibles. Se necesita el método setVisible para hacerlos visibles.
- Nacen con un tamaño inútil. Se necesita el método setSize para darles tamaño.
- Conviene decir qué debe hacer el programa si se cierra un frame.



Métodos importantes de JFrame



Práctica Propuesta1:

Crea un nuevo paquete que se llame practicasPropuestas.

PrimeraPracticaPropuesta.java

Crea una ventana, cuyo título sea Primera Practica Propuesta, define el tamaño 500,300 y la posición 200,300.

Creando un Marco Centrado



Clase Toolkit





```
graph LR; Toolkit[Toolkit] --> Methods[getDefaultToolkit()  
getScreenSize()];
```

<http://dis.um.es/~bmoros/Tutorial/parte13/cap13-12.html>

Práctica Propuesta2:

Dentro del paquete de practicasPropuestas.

SegundaPracticaPropuesta.java

Crea una ventana, cuyo título sea Segunda Practica Propuesta, define el tamaño y la posición con un solo método, tamaño 500,300 y posición 200, 300. Que la ventana no sea redimensionable. Cambia el icono de la ventana por el logotipo del ies.

Escibiendo en el Marco



Estructura de un JFrame

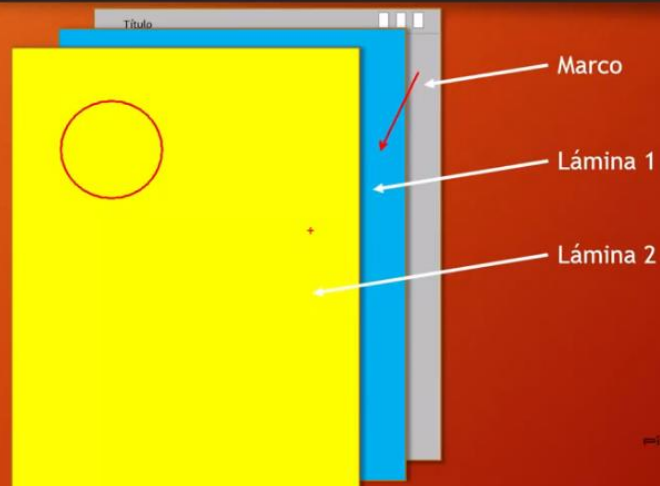




Marco



Estructura de un JFrame



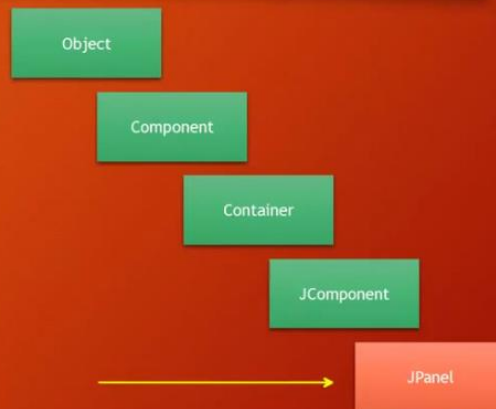
PILODRA INFORMÁTICAS



¿Cómo crear una lámina?



- La clase JPanel es la encargada de construir láminas donde poder dibujar y escribir. Debemos crear una clase que herede de JPanel



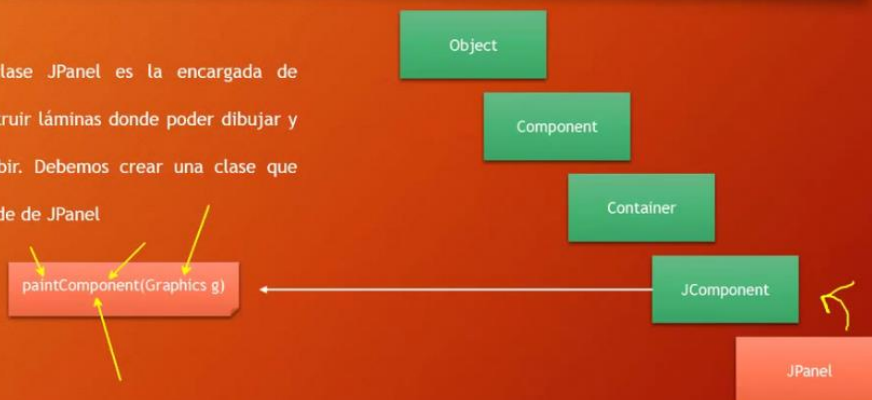
PILODRA INFORMÁTICAS



¿Cómo crear una lámina?




- La clase JPanel es la encargada de construir láminas donde poder dibujar y escribir. Debemos crear una clase que herede de JPanel




06:17

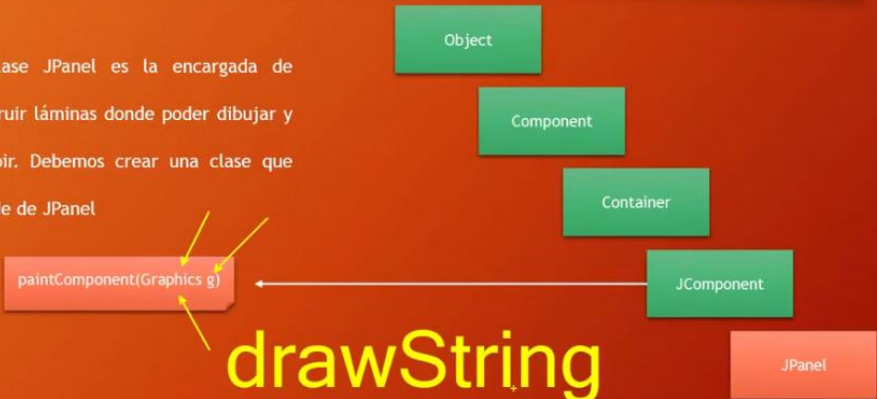
PILODRA INFORMÁTICAS



¿Cómo crear una lámina?



- La clase JPanel es la encargada de construir láminas donde poder dibujar y escribir. Debemos crear una clase que herede de JPanel



PILDORASINFORMÁTICAS

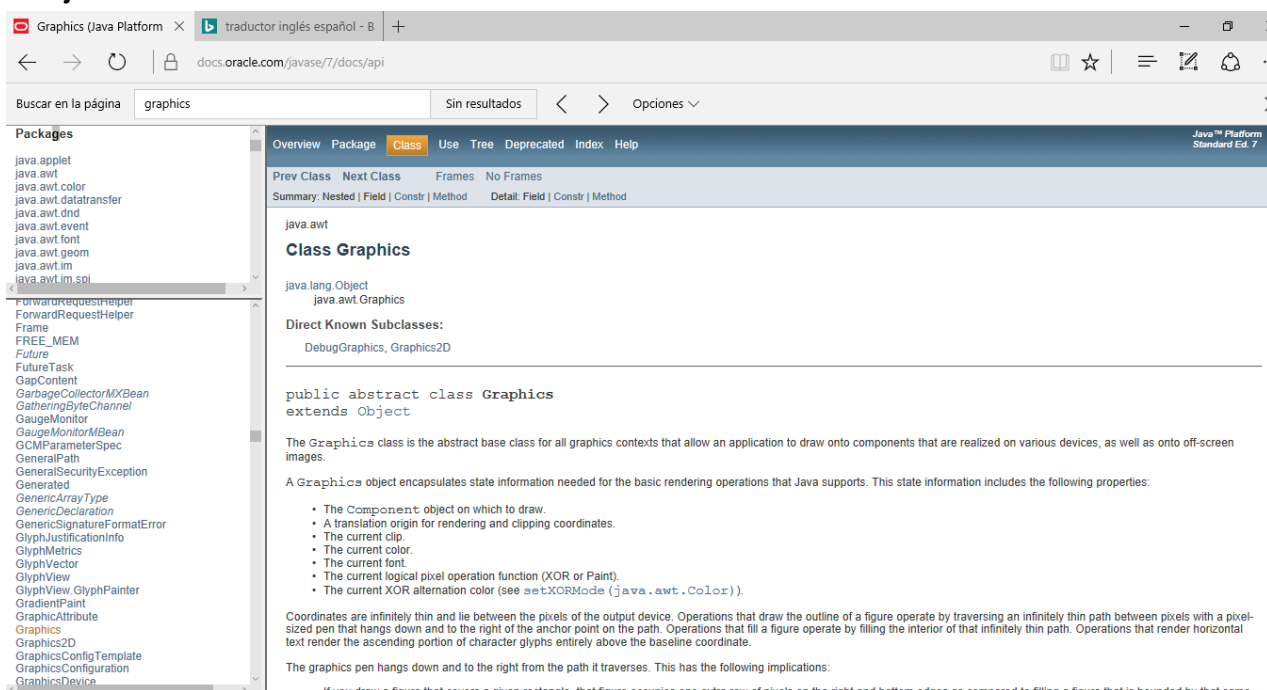
Práctica Propuesta3:

Dentro del paquete de practicasPropuestas.

TerceraPracticaPropuesta.java

Crea una ventana, en la cual aparecerá el texto “Cabecera” en la parte superior centrada y el texto “Pie” en la parte inferior derecha.

Dibujando en el Marco



Graphics (Java Platform) × traductor inglés español - B +

docs.oracle.com/javase/7/docs/api

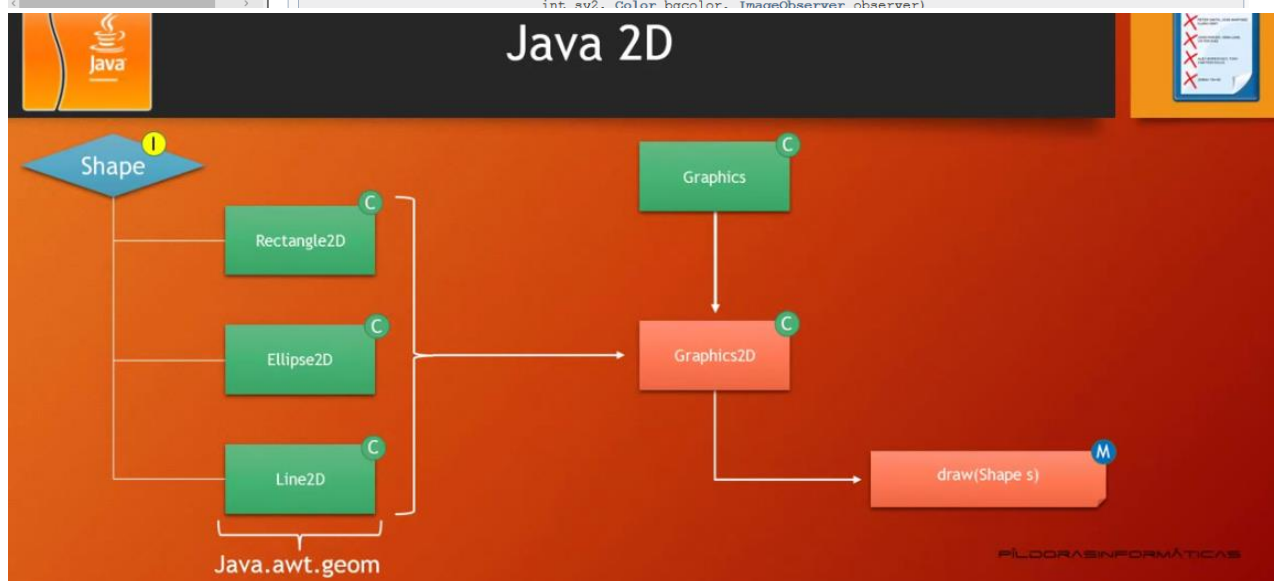
Buscar en la página graphics Sin resultados < > Opciones ▾

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- java.awt.dnd
- java.awt.event
- java.awt.font
- java.awt.geom
- java.awt.im
- java.awt.im.spl

ForwardRequestHelper
ForwardRequestHelper
Frame
FREE_MEM
Future
FutureTask
GapContent
GarbageCollectorMXBean
GatheringByteChannel
GaugeMonitor
GaugeMonitorMXBean
GCMParameterSpec
GeneralPath
GeneralSecurityException
Generated
GenericArrayType
GenericDeclaration
GenericSignatureFormatError
GlyphJustificationInfo
GlyphMetrics
GlyphVector
GlyphView
GlyphView, GlyphPainter
GradientPaint
GraphicsAttribute
Graphics
Graphics2D
GraphicsConfigTemplate
GraphicsConfiguration
GraphicsDevice

abstract void	<code>clearRect(int x, int y, int width, int height)</code> Clears the specified rectangle by filling it with the background color of the current drawing surface.
abstract void	<code>clipRect(int x, int y, int width, int height)</code> Intersects the current clip with the specified rectangle.
abstract void	<code>copyArea(int x, int y, int width, int height, int dx, int dy)</code> Copies an area of the component by a distance specified by dx and dy.
abstract Graphics	<code>create()</code> Creates a new Graphics object that is a copy of this Graphics object.
Graphics	<code>create(int x, int y, int width, int height)</code> Creates a new Graphics object based on this Graphics object, but with a new translation and clip area.
abstract void	<code>dispose()</code> Disposes of this graphics context and releases any system resources that it is using.
void	<code>draw3DRect(int x, int y, int width, int height, boolean raised)</code> Draws a 3-D highlighted outline of the specified rectangle.
abstract void	<code>drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code> Draws the outline of a circular or elliptical arc covering the specified rectangle.
void	<code>drawBytes(byte[] data, int offset, int length, int x, int y)</code> Draws the text given by the specified byte array, using this graphics context's current font and color.
void	<code>drawChars(char[] data, int offset, int length, int x, int y)</code> Draws the text given by the specified character array, using this graphics context's current font and color.
abstract boolean	<code>drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)</code> Draws as much of the specified image as is currently available.
abstract boolean	<code>drawImage(Image img, int x, int y, ImageObserver observer)</code> Draws as much of the specified image as is currently available.
abstract boolean	<code>drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)</code> Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.
abstract boolean	<code>drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)</code> Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.
abstract boolean	<code>drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor, ImageObserver observer)</code>



Manipular los dibujos. Manejando colores



Trabajando la fuente en un Frame.



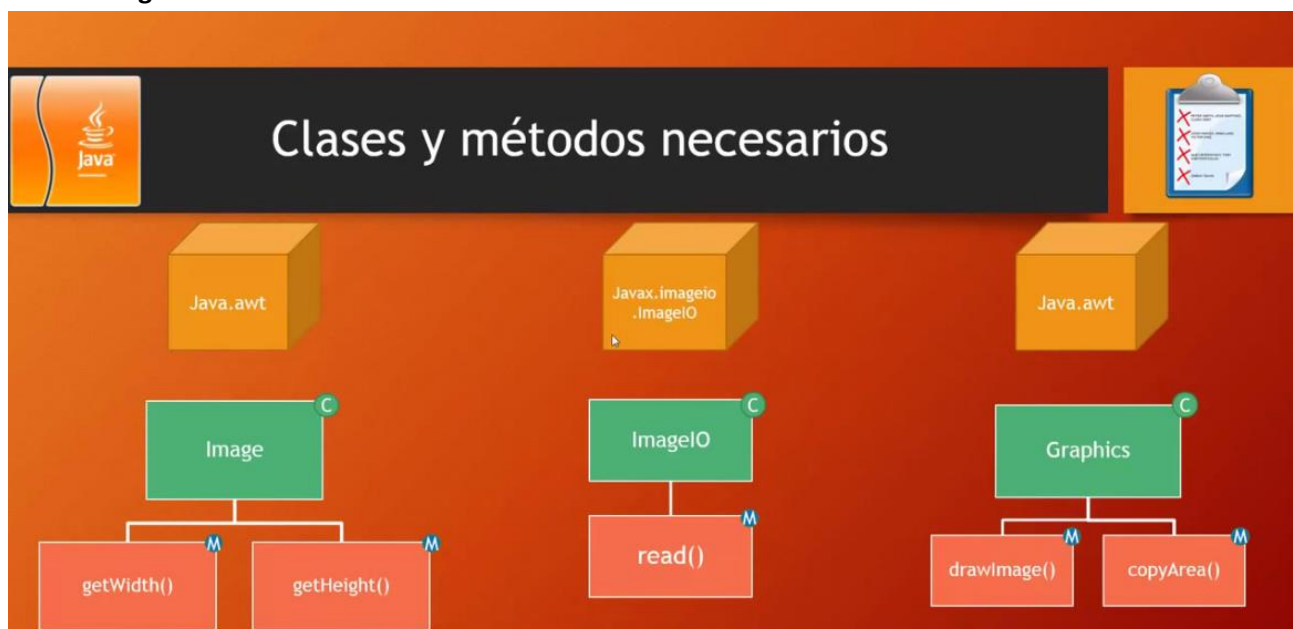
Práctica Propuesta4:

Dentro del paquete de practicasPropuestas.

CuartaPracticaPropuesta.java

Crea una ventana, en la que se muestre un cartel de prohibido utilizar móviles en clase. Crea el texto con un tipo de letra grande, visible y en un color destacado, crea un objeto tipo rectángulo y táchalo con una línea.

Incluir Imágenes en lo Frames



Práctica Propuesta5:

Dentro del paquete de practicasPropuestas.

QuintaPracticaPropuesta.java

Crea una ventana, en la que se muestre un cartel de prohibido utilizar móviles en clase. Crea el texto con un tipo de letra grande, visible y en un color destacado, inserta una imagen de un móvil e incorpórale un símbolo de prohibido superpuesto.

Eventos: Fuentes y Oyentes.

Las tres partes requeridas para el mecanismo de manejo de eventos son:

- El origen del evento.
- El objeto del evento y
- El componente de escucha del evento.

El origen del evento es el componente específico de la GUI con el que interactúa el usuario. El objeto del evento encapsula información acerca del evento que ocurrió, como una referencia al origen del evento, y cualquier información específica del evento que pueda requerir el componente de escucha del evento, para que pueda manejarlo. El componente de escucha del evento es un objeto que recibe una notificación del origen del evento cuando éste ocurre; en efecto, “escucha” un evento, y uno de sus métodos se ejecuta en respuesta al evento. Un método del componente de escucha del evento recibe un objeto evento cuando se notifica al componente de escucha acerca del evento. Después, el componente de escucha del evento utiliza el objeto evento para responder al evento. A este modelo de manejo de eventos se le conoce como modelo de eventos por delegación: el procesamiento de un evento se delega a un objeto específico (el componente de escucha de eventos) en la aplicación.

Para cada tipo de objeto evento, hay por lo general una interfaz de escucha de eventos que le corresponde. Cada interfaz de escucha de eventos especifica uno o más métodos manejadores de eventos que deben declararse en la clase que implementa a la interfaz

Oyente de un evento

Ya que sabemos que el oyente de un componente gráfico se encarga de monitorizarle y que en todo momento está a la escucha, de tal forma que, si se produce una interacción, responde de manera programada; y que el oyente es un objeto de una clase que implementa una interface de tipo Listener. Ahora conoceremos que hay diferentes listeners, que dependen de los componentes.

Los objetos que hacen de oyentes de un componente se crean de tres formas:

1. Un objeto de una clase que lo implemente.

```
class Oyente implements ActionListener {...}
```

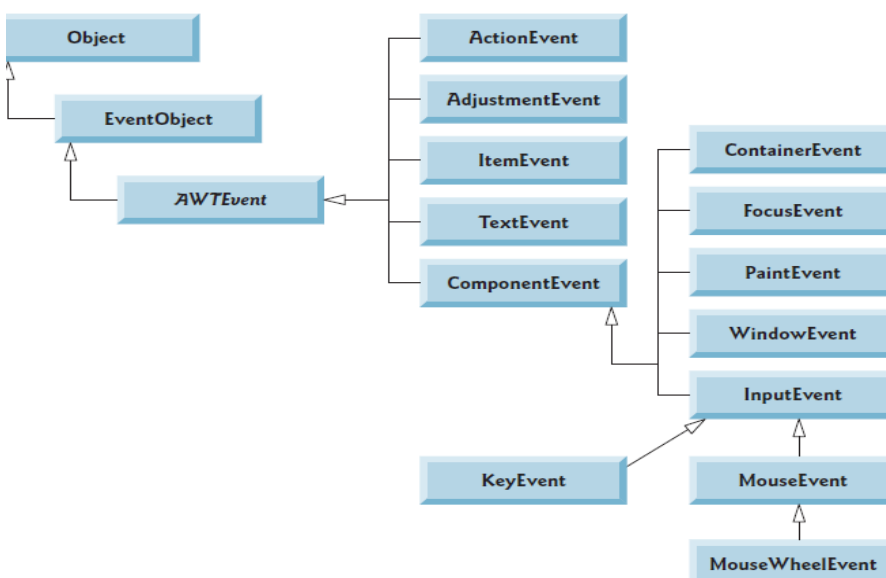
2. La clase que define al contenedor principal (marco, panel) lo declara e implementa.

```
class MarcoPpal extends JFrame implements ActionListener {...}
```

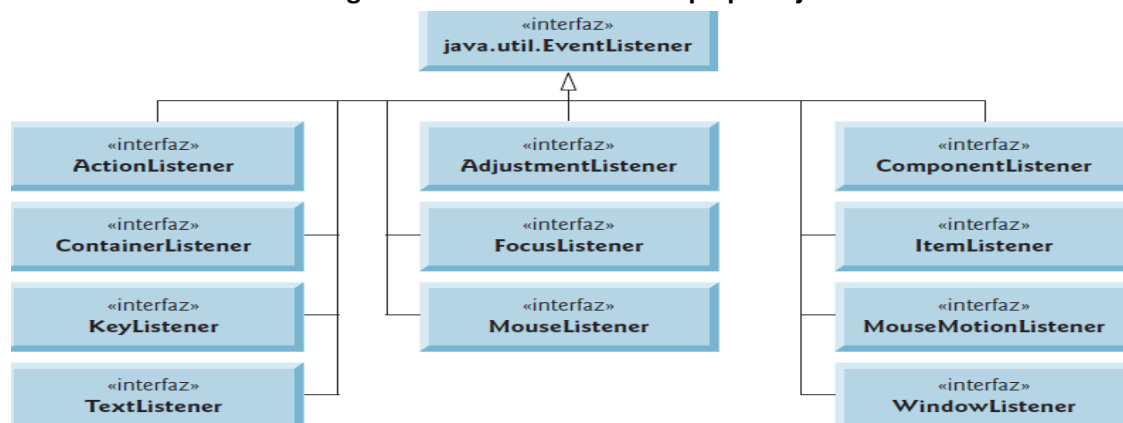
3. Directamente, creando un objeto anónimo que lo implemente.

```
new AdjustmentListener(){ ... }
```

Siempre será necesario registrar o asociar el objeto oyente al componente; para ello se llama al método `addxxxListener(oyente);`



Algunas clases de eventos del paquete java.awt.event.



Algunas interfaces comunes de componentes de escucha de eventos del paquete java.awt.event.

Componente	Eventos generados	Acción
JButton	ActionEvent	Hace clic en el botón.
JCheckBox	ItemEvent	Selecciona o deselecciona un ítem.
JCheckboxMenuItem	ItemEvent	Selecciona o deselecciona un ítem.
JDialog	WindowEvent	Actúa sobre ventanas de diálogo: abrir, cerrar, etcétera.
JRadioButton	ActionEvent	Hace clic en el botón de radio.
JList	ActionEvent	Hace doble clic sobre un ítem de la lista.
JList	ItemEvent	Selecciona o deselecciona un ítem de la lista.
JMenuItem	ActionEvent	Selecciona un ítem de un menú.
JScrollBar	AdjustementEvent	Cambia el valor de la scrollbar.
JTextComponent	TextEvent	Cambia el texto.
JTextField	ActionEvent	Termina de editar un texto pulsando Intro.
Window	WindowEvent	Actúa sobre una ventana: abrir, cerrar, iconizar, restablecer e iniciar el cierre.
Window	ComponentEvent	Mueve, cambia de tamaño, muestra u oculta un componente.
Window	FocusEvent	Obtiene o pierde el focus.
Window	KeyEvent	Pulsar o soltar una tecla.
Window	MouseEvent	Pulsar o soltar un botón del ratón; entrar o salir de un componente; mover o arrastrar el ratón.

Evento	Interfaz 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(), mouseDragged(), mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowActivated(), windowDeactivated(), windowClosed(), windowClosing(), windowIconified(), windowDeiconified(), windowOpened()

Package java.awt.event

Provides interfaces and classes for dealing with different types of events fired by AWT components.

See: Description

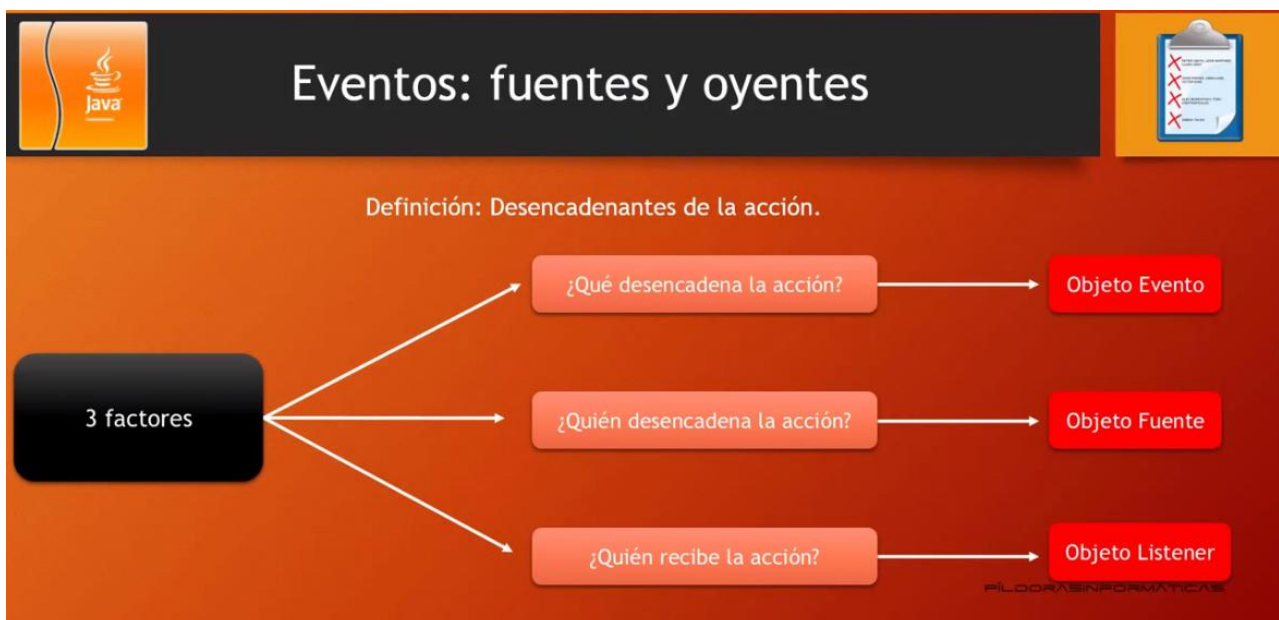
Interface Summary	
Interface	Description
ActionListener	The listener interface for receiving action events.
AdjustmentListener	The listener interface for receiving adjustment events.
AWTEventListener	The listener interface for receiving notification of events dispatched to objects that are instances of Component or MenuComponent or their subclasses.
ComponentListener	The listener interface for receiving component events.
ContainerListener	The listener interface for receiving container events.
FocusListener	The listener interface for receiving keyboard focus events on a component.
HierarchyBoundsListener	The listener interface for receiving ancestor moved and resized events.
HierarchyListener	The listener interface for receiving hierarchy changed events.
InputMethodListener	The listener interface for receiving input method events.
ItemListener	The listener interface for receiving item events.
KeyListener	The listener interface for receiving keyboard events (keystrokes).
MouseListener	The listener interface for receiving "interesting" mouse events (press, release, click, enter, and exit) on a component.
MouseMotionListener	The listener interface for receiving mouse motion events on a component.
MouseWheelListener	The listener interface for receiving mouse wheel events on a component.
TextListener	The listener interface for receiving text events.
WindowFocusListener	The listener interface for receiving WindowEvents, including WINDOW_GAINED_FOCUS and WINDOW_LOST_FOCUS events.
WindowListener	The listener interface for receiving window events.
WindowStateListener	The listener interface for receiving window state events.

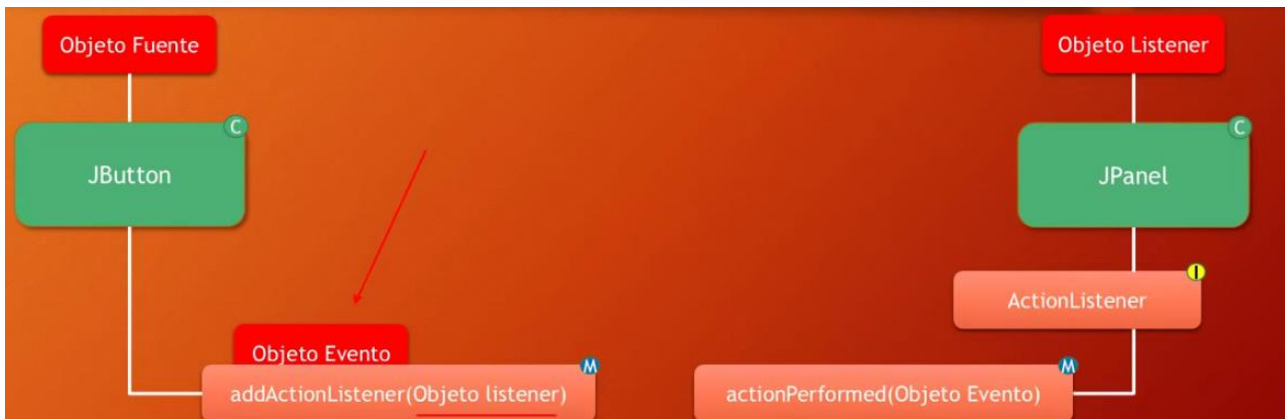
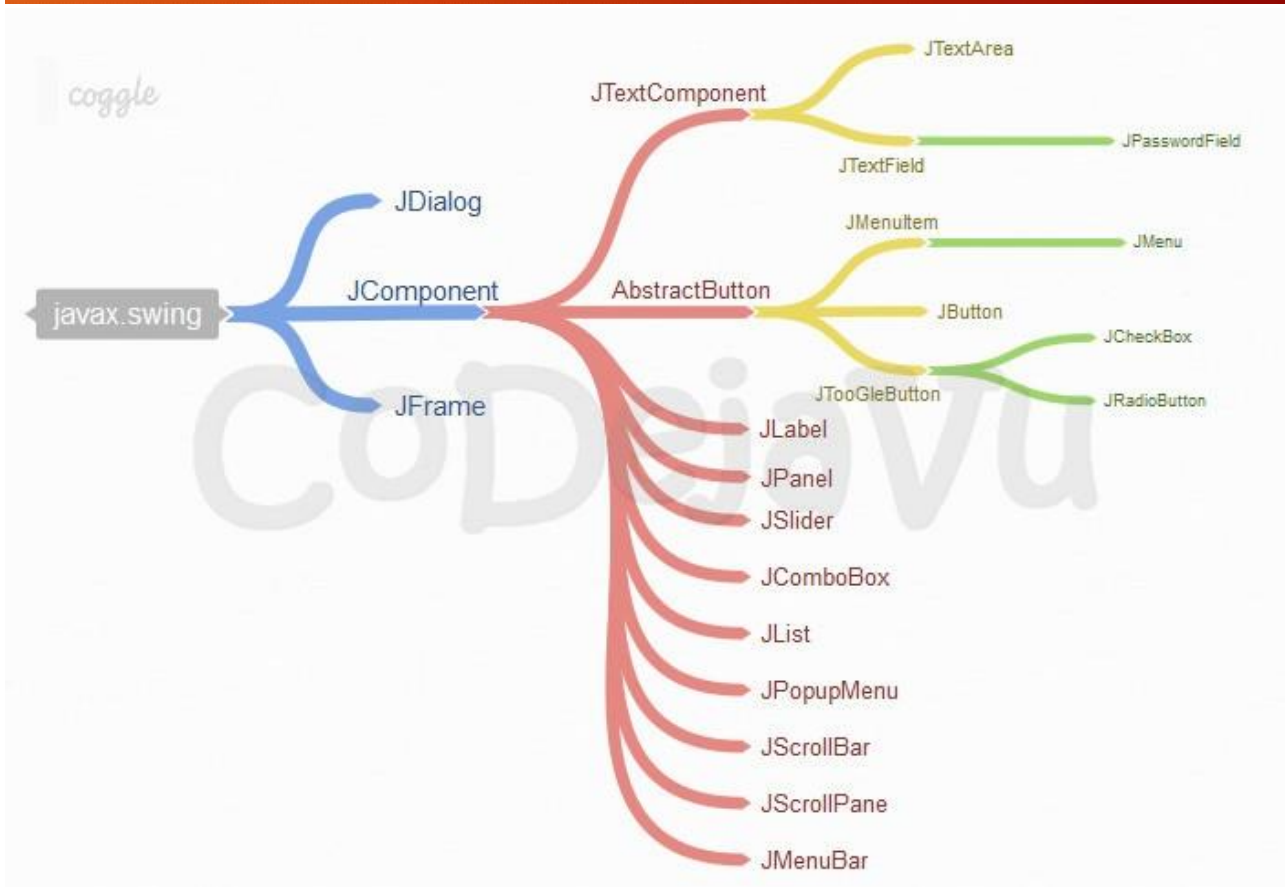
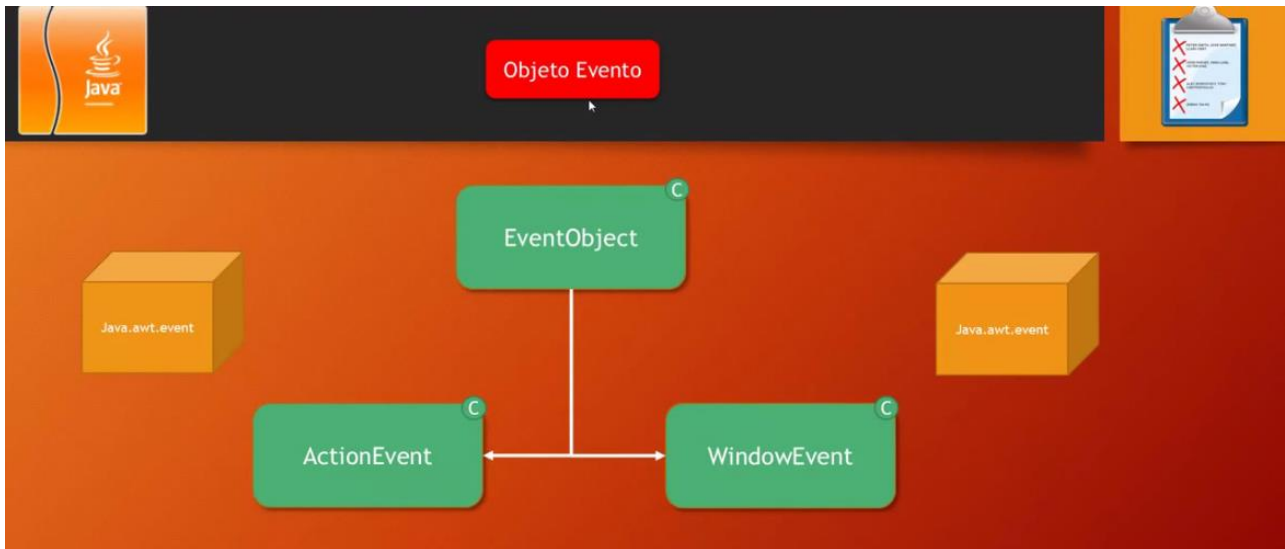
Class Summary	
Class	Description
ActionEvent	A semantic event which indicates that a component-defined action occurred.
AdjustmentEvent	The adjustment event emitted by Adjustable objects like Scrollbar and ScrollPane.
AWTEventListenerProxy	A class which extends the EventListenerProxy specifically for adding an AWTEventListener for a specific event mask.
ComponentAdapter	An abstract adapter class for receiving component events.
ComponentEvent	A low-level event which indicates that a component moved, changed size, or changed visibility (also, the root class for the other component-level events).
ContainerAdapter	An abstract adapter class for receiving container events.
ContainerEvent	A low-level event which indicates that a container's contents changed because a component was added or removed.
FocusAdapter	An abstract adapter class for receiving keyboard focus events.
FocusEvent	A low-level event which indicates that a Component has gained or lost the input focus.
HierarchyBoundsAdapter	An abstract adapter class for receiving ancestor moved and resized events.
HierarchyEvent	An event which indicates a change to the Component hierarchy to which Component belongs.
InputEvent	The root event class for all component-level input events.
InputMethodEvent	Input method events contain information about text that is being composed using an input method.
InvocationEvent	An event which executes the run() method on a Runnable when dispatched by the AWT event dispatcher thread.
ItemEvent	A semantic event which indicates that an item was selected or deselected.
KeyAdapter	An abstract adapter class for receiving keyboard events.
KeyEvent	An event which indicates that a keystroke occurred in a component.
MouseAdapter	An abstract adapter class for receiving mouse events.
MouseEvent	An event which indicates that a mouse action occurred in a component.
MouseMotionAdapter	An abstract adapter class for receiving mouse motion events.
MouseWheelEvent	An event which indicates that the mouse wheel was rotated in a component.
PaintEvent	The component-level paint event.
TextEvent	A semantic event which indicates that an object's text changed.
WindowAdapter	An abstract adapter class for receiving window events.
WindowEvent	A low-level event that indicates that a window has changed its status.

Package java.awt.event Description

Provides interfaces and classes for dealing with different types of events fired by AWT components. See the java.awt.AWTEvent class for details on the AWT event model. Events are fired by event sources. An event listener registers with an event source to receive notifications about the events of a particular type. This package defines events and event listeners, as well as event listener adapters, which are convenience classes to make easier the process of writing event listeners.

Since: 1.0






```

addActionListener

public void addActionListener(ActionListener l)

Adds the specified action listener to receive action events from this button. Action events occur when a user presses or releases the mouse over this button. If l is null, no exception is thrown and no action is performed.
Refer to AWT Threading Issues for details on AWT's threading model.

Parameters:
    l - the action listener

Since:
    JDK1.1

See Also:
    removeActionListener(java.awt.event.ActionListener), getActionListeners(), ActionListener

```

Se definen las acciones a realizar al producirse el evento sobre el componente.

MÉTODO 1: El Objeto Oyente es: Un objeto de una clase que lo implemente

`PruebaEventos.java`

MÉTODO 2: El Objeto Oyente es: La clase que define al contenedor principal (marco, panel) lo declara e implementa

`PruebaEventos2.java`

MÉTODO 3: La clase oyente: se implementa directamente, creando un objeto anónimo que lo implemente

`PruebaEventos3.java`

Práctica Propuesta6:

Dentro del paquete de practicasPropuestas.

`SextaPracticaPropuesta.java`

Crea una ventana, en la que se muestre un botón con el título “Saludo” y otro con el título “Despedida”, al pulsar el botón “Saludo” aparezca un mensaje(`JOptionPane.showMessageDialog`) que diga “HOLA” y al pulsar el botón “Despedida” ” aparezca un mensaje(`JOptionPane.showMessageDialog`) que diga “ADIOS”.

Práctica Propuesta6bis:

Dentro del paquete de practicasPropuestas.

`SextaBisPracticaPropuesta.java`

Realiza una modificación de la aplicación anterior de tal forma que los textos de “Hola” y “Adiós” se muestren en la propia ventana.

Eventos de ventana.



Hemos podido desarrollar todos los métodos menos `windowClosed` ya que este ya lo tenemos definido por defecto con `miMarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`

Si utilizamos la constante `EXIT_ON_CLOSE`, cuando cerramos una se finaliza el programa y por lo tanto se cierran las dos, entonces tendremos que utilizar en la segunda ventana otra constante `DISPOSE_ON_CLOSE`.



Si queremos desencadenar una acción cuando una ventana cambia de estado dandome igual cual sea este, tenemos que implementar todos los métodos de `WindowListener` o utilizar la clase adaptadora.

Si la clase oyente hereda de `WindowAdapter` por lo tanto hereda todos sus métodos, así solo tenemos que reescribir el método o métodos que nos interesen, es un truco de los programadores de java.

Controlando los estados de la ventana



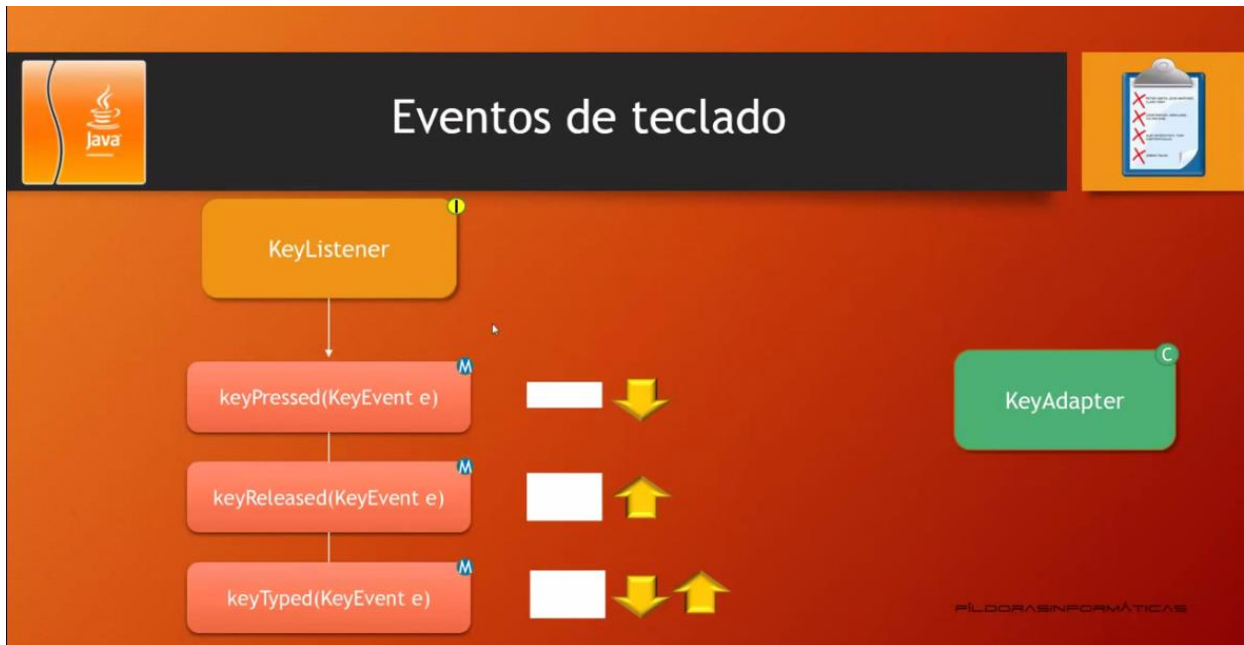
Práctica Propuesta7:

Dentro del paquete de practicasPropuestas.

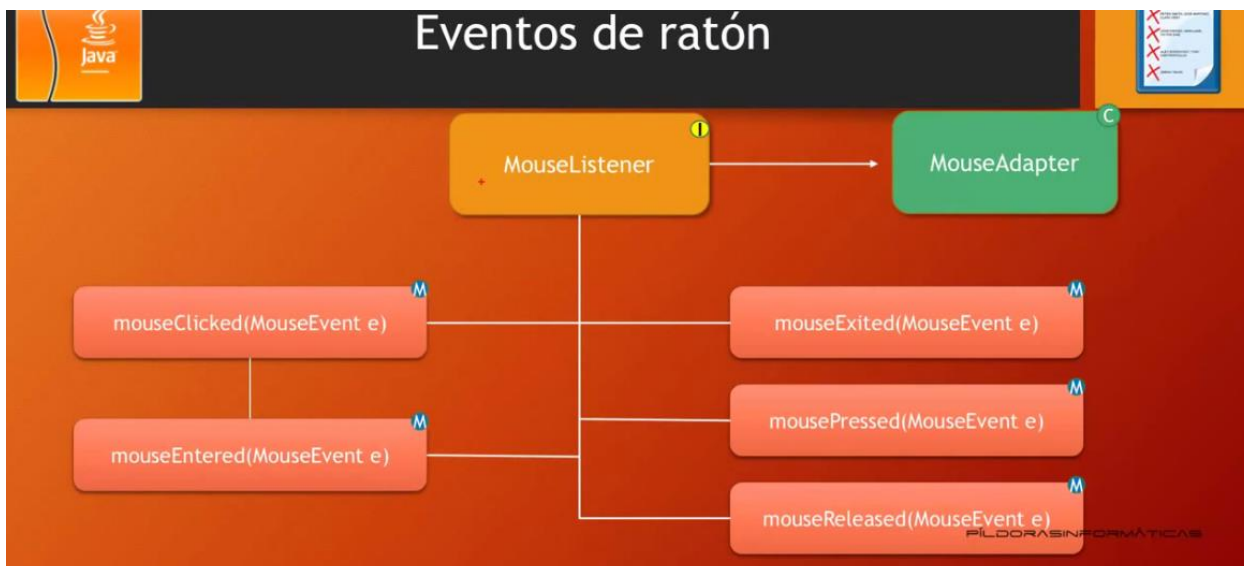
SeptimaPracticaPropuesta.java

Crea dos ventanas, las cuales les vaya cambiando el color de fondo según la ventana esté activada o desactivada.

Eventos de teclado



Eventos de ratón



De los tres primeros lo lógico es utilizar solo mouseClicked que cuando el clic se ha hecho y se ha soltado, para ello utilizamos la clase adaptadora



Práctica Propuesta8:

Dentro del paquete de practicasPropuestas.

OctavaPracticaPropuesta.java

Comprobar si el usuario ha hecho un click con el boton izquierdo ó con el derecho. La información aparezca por consola.

Buscar en la página: MouseMotionListener Sin resultados Opciones

Java™ Platform Standard Ed. 7

All Classes

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer

ModelBean

- ModelBeanAttributeInfo
- ModelBeanConstructorInfo
- ModelBeanInfo
- ModelBeanInfoSupport
- ModelBeanNotificationBroadcaster
- ModelBeanNotificationInfo
- ModelBeanOperationInfo
- ModificationItem
- Modifier
- Monitor
- MonitorInfo
- MonitorMBean
- MonitorNotification
- MonitorSettingException
- MouseAdapter
- MouseDragGestureRecognizer
- MouseEvent
- MouseInfo
- MouseListener
- MouseMotionAdapter
- MouseMotionListener
- MouseWheelEvent
- MouseWheelListener
- MTOM
- MTOMFeature

java.awt.event

Interface MouseMotionListener

All Superinterfaces:

- EventListener

All Known Subinterfaces:

- MouseListener

All Known Implementing Classes:

AWTEventMulticaster, BasicButtonListener, BasicComboPopup.InvocationMouseHandler, BasicComboPopup.InvocationMouseMotionHandler, BasicComboPopup.ListMouseHandler, BasicComboPopup.ListMouseMotionHandler, BasicDesktopIconUI.MouseInputHandler, BasicFileChooserUI.DoubleClickListener, BasicInternalFrameUI.BorderListener, BasicInternalFrameUI.GlassPaneDispatcher, BasicListUI.MouseInputHandler, BasicMenuItemUI.MouseInputHandler, BasicMenuUI.MouseInputHandler, BasicScrollBarUI.ArrowButtonListener, BasicScrollBarUI.TrackListener, BasicSliderUI.TrackListener, BasicSplitPaneDivider.MouseHandler, BasicTabbedPaneUI.MouseHandler, BasicTableHeaderUI.MouseInputHandler, BasicTableUI.MouseInputHandler, BasicTextUI.BasicCaret, BasicToolBarUI.DockingListener, BasicTreeUI.MouseHandler, BasicTreeUI.MouseInputHandler, DefaultCaret, FormView.MouseEventListener, HTMLEditorKit.LinkController, MetaFileChooserUI.SingleClickListener, MetaToolBarUI.MetalDockingListener, MouseAdapter, MouseDragGestureRecognizer, MouseInputAdapter, MouseMotionAdapter, ToolTipManager

```

public interface MouseMotionListener
extends EventListener

The listener interface for receiving mouse motion events on a component. (For clicks and other mouse events, use the MouseListener.)

The class that is interested in processing a mouse motion event either implements this interface (and all the methods it contains) or extends the abstract MouseMotionAdapter class (overriding only the methods of interest).

The listener object created from that class is then registered with a component using the component's addMouseMotionListener method. A mouse motion event is generated when the mouse is moved or dragged. (Many such events will be generated). When a mouse motion event occurs, the relevant method in the listener object is invoked, and the MouseEvent is passed to it.

Since:
    1.1

See Also:
    ...
  
```

Práctica Propuesta9:

Dentro del paquete de practicasPropuestas.

NovenaPracticaPropuesta.java

Comprobar si el usuario esta arrastrando el raton o simplemente lo está moviendo. La información aparezca por consola.

Eventos de foco componentes

El elemento que tiene el foco, es el que está seleccionado o donde estas posicionado, ejemplo la ventana activa, el cuadro de texto activo de un formulario, el elemento seleccionado de una lista de ficheros, etc



JLabel.

Son etiquetas de texto, sin embargo podemos usar sus propiedades para vincular imágenes por lo general las utilizamos para títulos, nombres o información puntual que queremos mostrar.

```
1 JLabel miLabel;  
2 miLabel= new JLabel();  
3 miLabel.setText("Esto es un Label");
```

Constructors
Constructor and Description
JLabel() Creates a JLabel instance with no image and with an empty string for the title.
JLabel(Icon image) Creates a JLabel instance with the specified image.
JLabel(Icon image, int horizontalAlignment) Creates a JLabel instance with the specified image and horizontal alignment.
JLabel(String text) Creates a JLabel instance with the specified text.
JLabel(String text, Icon icon, int horizontalAlignment) Creates a JLabel instance with the specified text, image, and horizontal alignment.
JLabel(String text, int horizontalAlignment) Creates a JLabel instance with the specified text and horizontal alignment.

JTextField.

Es uno de los componentes más comunes, permite introducir un campo de texto simple en nuestra ventana, ideal para introducir o mostrar datos puntuales en un formulario.

```
1 cajaDeTexto = new JTextField();  
2 cajaDeTexto.setText("Hola");  
3 cajaDeTexto.setBounds(90, 60, 90, 23);
```

Constructors
Constructor and Description
JTextField() Constructs a new TextField.
JTextField(Document doc, String text, int columns) Constructs a new JTextField that uses the given text storage model and the given number of columns.
JTextField(int columns) Constructs a new empty TextField with the specified number of columns.
JTextField(String text) Constructs a new TextField initialized with the specified text.
JTextField(String text, int columns) Constructs a new TextField initialized with the specified text and columns.

Methods inherited from class javax.swing.text.JTextComponent

```
addCaretListener, addInputMethodListener, addKeymap, copy, cut, fireCaretUpdate, getCaret, getCaretColor,
getCaretListeners, getCaretPosition, getDisabledTextColor, getDocument, getDragEnabled, getDropLocation,
getDropMode, getFocusAccelerator, getHighlighter, getInputMethodRequests, getKeymap, getMargin,
getNavigationFilter, getPreferredSize, getScrollableBlockIncrement, getScrollableUnitIncrement, getSelectedText,
getSelectedTextColor, getSelectionColor, getSelectionEnd, getSelectionStart, getText, getText, getToolTipText,
getUI, isEditable, loadKeymap, modelToView, moveCaretPosition, paste, print, print, print,
processInputMethodEvent, read, removeCaretListener, removeKeymap, removeNotify, replaceSelection,
restoreComposedText, saveComposedText, select, selectAll, setCaret, setCaretColor, setCaretPosition,
setComponentOrientation, setDisabledTextColor, setDragEnabled, setDropMode, setEditable, setFocusAccelerator,
setHighlighter, setKeymap, setMargin, setNavigationFilter, setSelectedTextColor, setSelectionColor,
setSelectionEnd, setSelectionStart, setText, setUI, updateUI, viewToModel, write
```

Eventos de foco ventanas



Práctica Propuesta10:

Dentro del paquete de practicasPropuestas.

DecimaPracticaPropuesta.java

Crea una aplicación que utilice un campo de texto normal y un campo de texto de contraseña. Cuando el usuario escribe un texto en uno de los campos y presiona la tecla intro, la aplicación debe mostrar un cuadro de diálogo de mensaje que contiene el texto que escribió el usuario.

Práctica Propuesta10bis:

Dentro del paquete de practicasPropuestas.

DecimaBisPracticaPropuesta.java

Realiza una modificación sobre la aplicación anterior para que los cuadros de diálogo de mensaje se muestren cuando el cuadro de texto pierda el foco.

Práctica Propuesta11:

Dentro del paquete de practicasPropuestas.

DecimoPrimeraPracticaPropuesta.java

Crea una ventana que contenga 2 botones, el boton "Curso de Java Inicial" y el boton "Curso de Java avanzado", al pulsar en cada uno de ellos en el título de la ventan aparacera el nombre del botón. Utiliza clases internas anónimas para implementar las interfaces de eventos.

Practica Propuesta12:

Dentro del paquete de practicasPropuestas.

DecimoSegundaPracticaPropuesta.java

Crea una calculadora que sume los dos valores introducidos en dos campos de texto, y el cálculo aparezca en un tercer campo en el momento que uno de los campos gane o pierdan el foco. Trata el posible error que se pueda generar al convertir String en Float.

Múltiples fuentes de eventos



Method Summary

Methods	
Modifier and Type	Method and Description
void	<code>addPropertyChangeListener(PropertyChangeListener listener)</code> Adds a PropertyChange listener.
Object	<code>getValue(String key)</code> Gets one of this object's properties using the associated key.
boolean	<code>isEnabled()</code> Returns the enabled state of the Action.
void	<code>putValue(String key, Object value)</code> Sets one of this object's properties using the associated key.
void	<code>removePropertyChangeListener(PropertyChangeListener listener)</code> Removes a PropertyChange listener.
void	<code>setEnabled(boolean b)</code> Sets the enabled state of the Action.

Methods inherited from interface <code>java.awt.event.ActionListener</code>	
	<code>actionPerformed</code>

¿Que métodos nos sirven para modificar las propiedades de un objeto fuente?

setEnabled(boolean); para poner activo o no un objeto.

isEnabled; nos dice si ese objeto esta activado o no.

¿Que métodos nos sirven para modificar las propiedades de un objeto oyente?

addPropertyChangeListener(PropertyChangeListener listener); añade una propiedad al oyente.

removePropertyChangeListener(PropertyChangeListener listener); elimina una propiedad del oyente.

Otras:

putValue(String key, Object value); almacena parejas de clave valor.

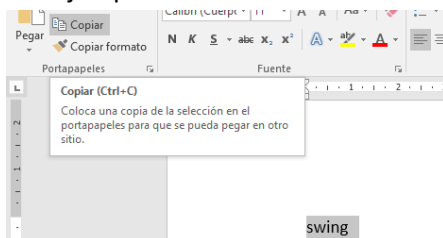
getValue(String key); recupera la clave almacenada previamente por putValue.

El ejemplo de la lamina con los tres botones(**PruebaEventos2**) que cambiaba de color según el botón que pulsáramos y además de le vamos a añadir que también cambiar de color con una combinación de teclas asociada. Es decir **dos** fuentes y **un** oyente.



En este caso el objeto fuente(evento) es capaz de guardar información del objeto como su nombre, icono, acciones, etc. `putValue` para guardar esa información, esta información viaja con el objeto `ActionEvent` evento, y podemos recuperarse esa información con `getValue`.

Para nuestro ejemplo vamos a almacenar: Nombre, Icono, Descripción y Acción.



Ejemplo: `swing`. Nombre: Copiar, Icono: Imagen con dos hojas, Descripción: el mensaje emergente (`setToolTipText`) y Acción: copiar.

Utilizamos `putValue(String key, Object value)`; almacena parejas de clave valor.

Necesitamos conocer los campos de clase **Action**.

Field Summary	
Fields	
Modifier and Type	Field and Description
static String	<code>ACCELERATOR_KEY</code> The key used for storing a <code>KeyStroke</code> to be used as the accelerator for the action.
static String	<code>ACTION_COMMAND_KEY</code> The key used to determine the command String for the <code>ActionEvent</code> that will be created when an <code>Action</code> is going to be notified as the result of residing in a <code>Keymap</code> associated with a <code>JComponent</code> .
static String	<code>DEFAULT</code> Not currently used.
static String	<code>DISPLAYED_MNEMONIC_INDEX_KEY</code> The key used for storing an <code>Integer</code> that corresponds to the index in the text (identified by the <code>NAME</code> property) that the decoration for a mnemonic should be rendered at.
static String	<code>LARGE_ICON_KEY</code> The key used for storing an <code>Icon</code> .
static String	<code>LONG_DESCRIPTION</code> The key used for storing a longer <code>String</code> description for the action, could be used for context-sensitive help.
static String	<code>MNEMONIC_KEY</code> The key used for storing an <code>Integer</code> that corresponds to one of the <code>KeyEvent</code> key codes.
static String	<code>NAME</code> The key used for storing the <code>String</code> name for the action, used for a menu or button.
static String	<code>SELECTED_KEY</code> The key used for storing a <code>Boolean</code> that corresponds to the selected state.
static String	<code>SHORT_DESCRIPTION</code> The key used for storing a short <code>String</code> description for the action, used for tooltip text.
static String	<code>SMALL_ICON</code> The key used for storing a small <code>Icon</code> , such as <code>ImageIcon</code> .

//1º paso. clase oyente

```

private class AccionColor extends AbstractAction{
    public AccionColor(String nombre, Icon icono, Color colorBoton){
        putValue(Action.NAME, nombre);
        putValue(Action.SMALL_ICON, icono);
        putValue(Action.SHORT_DESCRIPTION, "Poner la lámina de color " +
nombre);

        //también podemos crear nuestras parejas
        putValue("color_de_fondo", colorBoton);
    }

    public void actionPerformed(ActionEvent e) {
        //hacemos casting para que no de error
        Color c=(Color)getValue("color_de_fondo");
        //este metodo no corresponde a esta clase sino a JPanel, asi
convertimos en una clase
        //interna de JPanel
        setBackground(c);
    }
}
}

```

AccionColor **accionAmarillo**=new AccionColor("Amarillo",new ImageIcon("/src/graficos/amarillo.gif"), Color.YELLOW);

Constructor Summary

Constructors

Constructor and Description

JButton()

Creates a button with no set text or icon.

JButton(Action a)

Creates a button where properties are taken from the Action supplied.

JButton(Icon icon)

Creates a button with an icon.

JButton(String text)

Creates a button with text.

JButton(String text, Icon icon)

Creates a button with initial text and an icon.

add(new JButton(accionAmarillo));

Para recuperar valores de clave,valor:

Color c=(Color)getValue("color_de_fondo");



Asignando acciones a teclado

1. Crear mapa de entrada.
2. Crear combinación de teclas
3. Asignar combinación de teclas a objeto.
4. Asignar objeto a acción.

1. Crear mapa de entrada.
 - ¿Quién tiene el foco?

```

graph TD
    IM[InputMap]
    JC[JComponent] --- M["getInputMap(int condición)"]
    
```

InputMap **mapaEntrada**=getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW);

WHEN_FOCUSED <pre>public static final int WHEN_FOCUSED</pre> <p>Constant used for <code>registerKeyboardAction</code> that means that the command should be invoked when the component has the focus.</p> <p>See Also:</p> <p>Constant Field Values</p>
WHEN_ANCESTOR_OF_FOCUSED_COMPONENT + <pre>public static final int WHEN_ANCESTOR_OF_FOCUSED_COMPONENT</pre> <p>Constant used for <code>registerKeyboardAction</code> that means that the command should be invoked when the receiving component is an ancestor of the focused component or is itself the focused component.</p> <p>See Also:</p> <p>Constant Field Values</p>
WHEN_IN_FOCUSED_WINDOW <pre>public static final int WHEN_IN_FOCUSED_WINDOW</pre> <p>Constant used for <code>registerKeyboardAction</code> that means that the command should be invoked when the receiving component is in the window that has the focus or is itself the focused component.</p> <p>See Also:</p> <p>Constant Field Values</p>

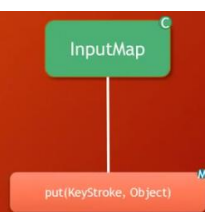
2. Crear combinación de teclas.

- Guardar descripción de teclas



KeyStroke **teclaAmarillo**=KeyStroke.getKeyStroke("ctrl A");

3. Asignar combinación de teclas a objetos.



mapaEntrada.put(teclaAmarillo, "fondo_amarillo");

//paso 2 y 3 juntos

mapaEntrada.put(KeyStroke.getKeyStroke("ctrl A"), "fondo_amarillo");

4. Asignar objetos a acciones.



ActionMap **mapaAccion**=getActionMap();

mapaAccion.put("fondo_amarillo", accionAmarillo);

Multiples oyentes



El ejemplo que vamos a hacer es de una fuente y varios oyentes.

Vamos a crear un programa con dos botones, el primer boton nos va a crear nuevos marcos y el segundo boton “cerrar todo”, va a cerrar todos los nuevos marcos que hemos creado, tenemos una fuente=“el boton cerrar todo” y varios oyentes que son todos los marcos.

```
dispose

public void dispose()

Releases all of the native screen resources used by this Window, its subcomponents, and all of its owned children. That is, the resources for these Components will be destroyed, any memory they consume will be returned to the OS, and they will be marked as undisplayable.

The Window and its subcomponents can be made displayable again by rebuilding the native resources with a subsequent call to pack or show. The states of the recreated Window and its subcomponents will be identical to the states of these objects at the point where the Window was disposed (not accounting for additional modifications between those actions).

Note: When the last displayable window within the Java virtual machine (VM) is disposed of, the VM may terminate. See AWT Threading Issues for more information.

See Also:
Component.isDisplayable(), pack(), show()
```

Como el fuente y el oyente estan en clases distintas, vamos a tener que pasar por parametro el objeto fuente a la clase que contiene al oyente.



Practica Propuesta13:

Dentro del paquete de practicasPropuestas.

DecimoTerceraPracticaPropuesta.java

Crea una calculadora que sume los dos valores introducidos en dos campos de texto y ponga el resultado en un tercer campo de texto al hacer clic sobre un botón llamado suma y cuando se presione la combinación de teclas “ctrl + s”. Indica con un cuadro emergente la función del botón. Incluye en el boton un icono. Trata el posible error que se pueda generar al convertir String en Float.