**SWING**

package graficos

1. CreandoMarcos

2. CreandoMarcoCentrado

3. EscribiendoEnMarco

4. PruebaDibujo

5. TrabajandoColores

6. TrabajandoConFuentes

6.1 ComprobarFuentesDelSistema

7. PruebaImagenes

**–EVENTOS**

8. PruebaEventos

9. PruebaEventos2

10. PruebaEventosVentanas: WindowListener (7 métodos)

11. PruebaEventosVentanas2: AdapterClasses(WindowAdapter)

12. CambioEstado (Control de ventanas: windowEvent)

13. EventosTeclado

14. EventosRaton

15. EventosFoco

16. EventosFocoVentana

17. EventosFuentes\_Acciones

18. EventosVariosOyentes

19. Layaouts

20. Layaout\_Grid\_Calculadora

**–COMPONENTES SWING**

21. CuadroTexto

22. EventosCuadroTexto

23. EventosCuadroTexto\_password

24. AreaTexto

25. CheckBoxEjemplo

26. RadioButtonEjemplo

27. RadioButtonEjemplo2

28. ComboBoxEjemplo

29. SlideEjemplo

30. SlideEjemplo2

31. MenuCreacion

\*\*

33. MarcoMenuConImagen

34. MarcoMenuConCheckBox\_RadioButton

35. MarcoMEmergente

**package practicaProcesadorTexto**

32. Procesador

**GLOSARIO**

(i) → interfaz

(m) → método

© → clase

***POSICIONAR VENTANA EN EL CENTRO DE CUALQUIER PANTALLA***

**toolkit** © → Almacén de multitud de métodos que se comunican con el sistema huésped de ventanas

* getDefaultTookit() (m)
* getScreenSize() (m)

***LÁMINAS***

**JPanel ©**→ clase encargada de construir láminas donde poder dibujar y escribir.

-Debemos crear una clase que herede de JPanel.

* paintComponent(Graphics g) --> es de JComponent una clase superior a JPanel

***DIBUJOS***

**Graphics** se ha ido quedando obsoleto

-Biblioteca **Graphics2D** que hereda de Graphics:

-Rectangle2D

-Ellipse2S

-Line2D

TODAS ESTAS IMPLEMENTAN UNA INTERFAZ LLAMADA: Shape (i)

-draw(Shape s) (m)

***COLORES***

**Graphics 2D** → setPaint(Color) (m)

**JPanel**

* setBackground(Color) (m)
* setForegound(Color) (m)

***FUENTES***

**Graphics 2D** → setFont(Color) 8m)

**Font** ©

* Font (tipo, estilo, tamaño) (m)
  + bold, plain, italic

\* Cada usuario tiene en su sistema sus fuentes descargadas por tanto a veces no es compatible la fuente que elegimos

***IMAGENES***

**\*Paquete: Java.awt**

**Image ©**

* getWidth() (m)
* getHeight() (m)

**\* Paquete: javax.imageio.\* y java.io.\***

**ImageIO ©**--> read() obtener la imagen de una carpeta o de un link.

**\* Paquete: Java.awt**

**\* Graphics ©**

* drawImage() (m)
* copyArea() (m)

**EVENTOS: fuentes y oyentes**

definición: desencadenantes de la acción

3 factores:

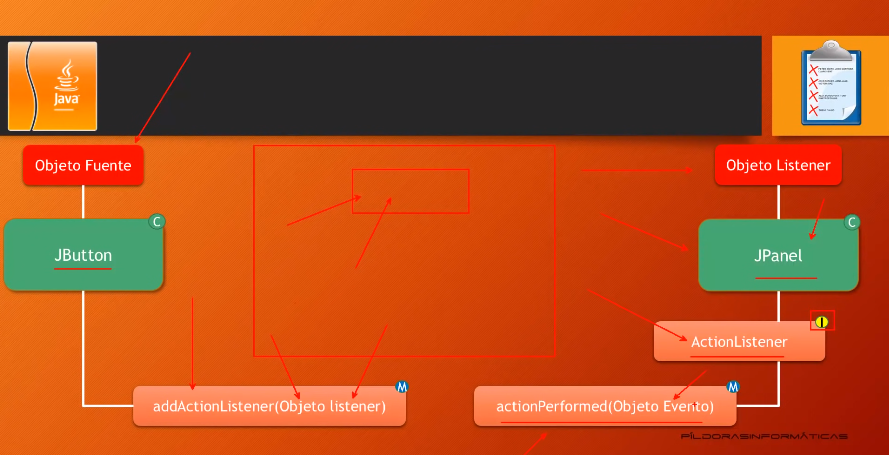
-¿Qué desencadena la acción?--> **Objeto Evento**

* **EventObject**
  + **ActionEvent:** ratón (java.awt.event)
  + **WindowEvent:** ventana (java.awt.event)

-¿Quién desencadena la acción?--> **Objeto Fuente**

-¿Quién recibe la acción?--> **Objeto Listener**

Ej word: al hacer click - ficha inicio - cinta de opciones (cambio de aspecto)



**EVENTOS VENTANA**

WindowListener: te obliga a declarar estos 7 métodos.

* **windowActivated**(WindowEvent e)
* **windowClosed**(WindowEvent e)
* **windowClosing**(WindowEvent e)
* **windowDeactivated**(WindowEvent e)
* **windowDeiconified**(WindowEvent e)
* **windowIconified**(WindowEvent e)
* **windowOpened**(WindowEvent e)

***\*ADAPTER CLASSES***

Solución a la implementación de 7 métodos. Limpieza y optimización de código.

**KeyAdapter**

**WindowAdapter**

* WindowFocusListener(i)-WindowStateListener(i)
* WindowListener(i)-EventListener(i)

**MouseAdapter**

***\*CONTROLANDO ESTADO VENTANA***

**WindowStateListener (i)→** windowStateChanged(WindowEvent e)

**WindowEvent ©**

* getNewState() (m)
* getOldState() (m)
* devuelven un entero. Podemos hacer comparaciónes con: Frame.estadodeventana

***\*EVENTOS DE TECLADO***

**KeyListener (i)**

* keyPressed(KeyEvent e): (m) cuando presionas la tecla
* keyReleased (KeyEvent e): (m) cuando dejes de presionar la tecla
* keyTyped(KeyEvent e): (m) tecla presionada y ha dejado de ser presionada (Combinación)
* vk\_0, vk\_s… → valores constantes de teclas

**KeyEvent ©**

* getKeyCode() (m) → obtener el código de la tecla
* getKeyChar() (m)→ obtener la letra

**KeyAdapter ©**

***\*EVENTOS DE RATÓN***

**MouseListener (i)** → MouseAdapter © → si solo queremos un método deberemos heredar está clase en vez de implementar la interfaz.

* mousedPressed(MouseEvent e) (m)
* mousedReleased(MouseEvent e) (m)
* mousedClicked(MouseEvent e) (m)
* mousedEntered(MouseEvent e) (m)
* mousedExisted(MouseEvent e) (m)

**MouseEvent ©**

* Constantes para saber el botón utilizado:
  + BUTTON1\_DOWN\_MASK → botón izquierdo
  + BUTTON2\_DOWN\_MASK → rueda
  + BUTTON3\_DOWN\_MASK → botón derecho
* getX() (m)→ posición del ratón
* getY() (m)
* getModifiersEx() (m)→ si se ha usado boton derecho, izquierdo o rueda
* getClickCount() (m)→ cuántos clics ha hecho el usuario

**MouseMotionListener (i)**

* mouseDragged() (m) → arrastrar
* mouseMoved() (m) →mover

***\*EVENTOS DE FOCO***

**FocusListener (i) →** FocusAdapter ©

* focusGained(Focusevent e) (m)
* focusLost(FocusEvent e) (m)

En el caso de que manejemos eventos con ventanas:

**windowsFocusListener(i) →**WindowAdapter ©

* windowGainedFocus(Focusevent e)
* windowLostFocus(FocusEvent e)

***\*EVENTOS: MÚLTIPLES FUENTES***

**Action (i) →** hereda → actionPerformed(ActionEvent evento) (m)

* setEnabled(boolean b) (m) → activar/desactivar
* isEnabled(boolean b) (m) → preguntar por el estado
* putValue(String clave, Object valor) (m) → almacenar pares clave, valor
* getValue(String clave, Object valor) (m)
* addPropertyChangedListener(PropertyChangeListener oyente) (m)
* removePropertyChangedListed (PropertyChangeListener oyente) (m)

**AbstractAction © →** heredamos de aquí para obtener todos estos métodos anteriores, y así no tienes que escribirlos.



**ASIGNANDO ACCIONES A TECLADO:**

* **KeyStroke ©** 
  + static getKeyStroke(String s) (m)
* **InputMap ©** 
  + put (KeyStroke, Object) (m)
* **JPanel ©** 
  + getInputMap(int condición) (m)
* **ActionMap ©** 
  + put (Object, acción) (m)

1)Crear mapa de entrada: ¿cuál va a ser el objeto en el que tiene que incidir la entrada?

2)Crear combinación de teclas

3)Asignar combinación de teclas a objeto

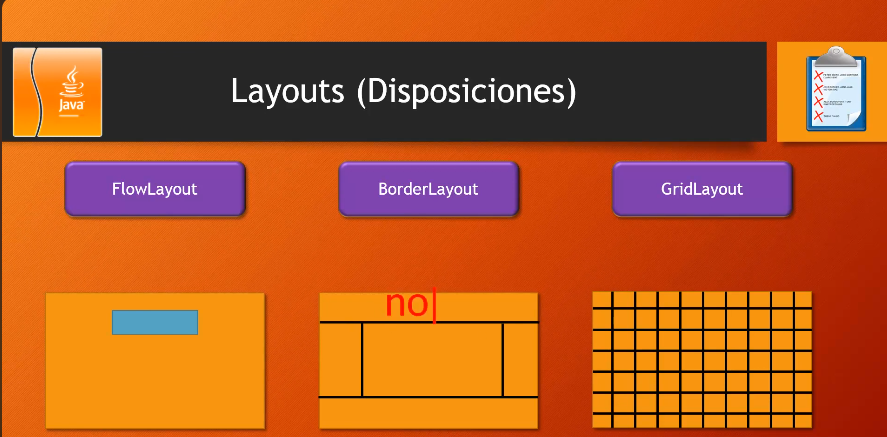
4)Asignar objeto a acción

***\*MULTIPLES OYENTES***

Hasta ahora hemos visto esto:



***\*LAYOUTS/DISPOSICIONES***

******

FlowLayout © → por defecto se usa este con la posición center.

BorderLayout ©

GridLayout (rejillas-celdas) ©

Container © → setLayout (m)

**COMPONENTES SWING**

***\*CUADROS DE TEXTO***

**JTextComponent (A) ©**

* void setText(String t) (m)
* void getText(String t) (m)
* **JTextField © →** construir cuadro de texto de una línea
* **JTextArea ©→** area de texto

**JLabel →** para poner texto

**–CAMBIOS EN UN CAMPO DE TEXTO:**

**JTextComponent (A) ©**

* **JTextField © →** construir cuadro de texto de una línea
* getDocument() (m) → Document (i) → addDocumentListener (m) → documentEvent() (e)
  + ReceptoraEvento ©→ DocumentListener (i)
    - removeUpdate()
    - insertUpdate()
    - changedUpdate()

–ÁREAS DE TEXTO

* **JTextArea ©**
  + **getText() (m)**
  + **setLineWrap(boolean) (m) →** saltos de línea.
  + **getLineWrap() (m)**

–CHECKBOX

* JCheckBox ©
  + isSelected() (m)
  + setSelected(boolean) (m)

–RADIO BUTTON

* JRadioButton ©
  + crear cada botón
* ButtonGroup ©
  + agrupar para que tengan el mismo funcionamiento

–COMBOBOX

* JComboBox ©
  + addItem() (m) → agregar elementos al desplegable
  + getSelectedItem() (m)→ devolver el elemento seleccionado

–SLIDER

* JSlider ©
  + setPaintTicks(boolean) (m) → imprimir lineas de division
  + setMajorTickSpacing(int) (m) → imprimir números
  + setMinorTickSpacing(int) (m) → decir cuanto espaciado
  + setPaintLabels(boolean) (m)

–SPINER

* JSpinner ©
  + Jspinner()
  + Jspinner(SpinnerModel modelo)

–CONSTRUCCIÓN MENÚS

* JMenuBar © → menú principal
* JMenu © → cada uno de los elementos el menú principal
* JMenuItem © → items de cada elemento del menú principal
  + JMenuItem(String texto) (m)
  + JMenuItem(String texto, icon icono) (m)→ con imagen
  + JCheckBoxMenuItem ©
  + JRadioButtonMenuItem ©
  + setAccelerator(KeyStroke obj) (m)
    - llamas pasando (KeyEvent.VK\_D, InputEvent.CTRL\_DOWN\_MASK) →constante (en este caso la D) + control.
* SEPARADORES: addSeparator() (m)
* JPopupMenu © → Menús emergentes
  + setComponentPopupMenu(Component c)

– BARRA DE HERRAMIENTAS (AVANZADO) → vídeos 111-114.

package graficos;

import javax.swing.\*;

import java.awt.\*;

public class Layouts {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

}

}

class Marco\_Layout extends JFrame {

public Marco\_Layout() {

setTitle("Prueba Acciones");

setBounds(600, 350, 600, 300);

Panel\_Layout lamina = new Panel\_Layout();

add(lamina);

}

}

class Panel\_Layout extends JPanel{

public Panel\_Layout() {

add(new JButton("Amarillo"));

add (new JButton("Rojo"));

add (new JButton ("Azul"));

}

}

package graficos;

import javax.swing.\*;

import java.awt.\*;

public class EventosCuadroTexto {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

MarcoPrueba mimarco = new MarcoPrueba();

mimarco.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

}

}

class MarcoPrueba extends JFrame{

public MarcoPrueba() {

setBounds(500,300,500,350);

LaminaPrueba milamina = new LaminaPrueba();

add(milamina);

setVisible(true);

}

}

class LaminaPrueba extends JPanel{

public LaminaPrueba() {

JTextField micampo = new JTextField(20);

add(micampo);

}

}