# 1ºEVALUACIÓN

## Eventos y componente JBtutton

-En java, cada evento (acción) está representado por un objeto, que es una subclase de ***EventObject.*** Cada ***EventObject*** representa un evento particular, por ejemplo:

* **MouseEvent** 🡪 acciones del ratón : mover arrastrar, click, etc.
* **KeyEvent** 🡪 acciones del teclado: pulsar teclas.
* **ActionEvent** 🡪 acción del usuario en la interfaz: pulsar un botón en la pantalla

-Los **EventObject** no realizan acciones por si mismos, sino que esos eventos son enviados a otro objeto encargado de responder a un evento en particular. Estos objetos son los que conocemos como ***listeners*** , y hay diferentes **listeners** que escuchan a diferentes eventos ( como el MouseListener, KeyListener,AcionListener,….).

-Los **Listener** se implementan como **interfaces**. Una ***interface*** es una colección de métodos que definen un comportamiento. Cualquier clase para estos métodos puede declarar dicha **interface.**  Una misma clase puede implementar varios interfaces a la vez, pero solo puede heredar una.

-Los listeners que usaremos se encuentran en el paquete **java.awt.event** ( que hay que importar) y algunos son:

### ActionListener

Se produce al hacer click en un componente (también si damos al enter)

#### **Metodo**

Public void ActionPerfomed(ActionEvent e)

#### Eventos

**JButton** 🡪 click o pulsar enter enfocado en el

**JList** 🡪 doble click en un elemento de la lista

**JMenultem** 🡪 seleccionar una opción del menú

**JTeztField** 🡪 al pulsar Enter con el foco activado

### KeyListener

Se produce al pulsar una tecla (puede variar la forma de pulsar la tecla)

#### Metodos

Public void KeyTyped(KeyEvent e)

Public void KeyPressed (KeyEvent e)

Public void KeyReleased (KeyEvent e)

#### Eventos (cuando pulsamos una tecla)

**KeyTyped** 🡪 pulsar y soltar

**KeyPressed** 🡪 pulsar

**KeyReleased** 🡪 soltar

### FocusListener

Se produce cuando un componente es o deja de estar seleccionado

#### Metodos

Recibir o perder el foco

### MouseListener

Se produce cuando realizamos una acción con el ratón.

#### Metodos

public void mouseClicked(MouseEvent e)  
public void mouseEntered(MouseEvent e)  
public void mouseExited(MouseEvent e)  
public void mousePressed(MouseEvent e)  
public void mouseReleased(MouseEvent e)

#### Eventos (según el listener)

**mouseClicked** 🡪 pinchar y soltar

**mouseEntered** 🡪 entrar en un componente con el puntero (pasar por encima)

**mouseExited** 🡪 salir de un componente con el puntero

**mousePressed** 🡪 presionar un botón

**mouseReleased** 🡪 soltar el botón

### MouseMotionListener

Se produce con el movimiento del mouse

#### Metodos

public void mouseDragged(MouseEvent e)  
public void mouseMoved(MouseEvent e)

#### Eventos (según el listener)

**mouseDraged** 🡪 click y arrastrar a un componente

**mouseMoved** 🡪 al mover el puntero sobre un elemento

### Invocar Listener

-Se puede hacer de esta manera:

import java.awt.event.\*;

boton1.addActionListener(new ActionListener(){

public void actionPerformed (){

//Acciones

}

});

-O de esta manera :

import java.awt.event.\*;

ActionListener al=new ActionListener(){

public void actionPerformed (){

//Acciones

}

};

boton1.addActionListener(al);

### JButton

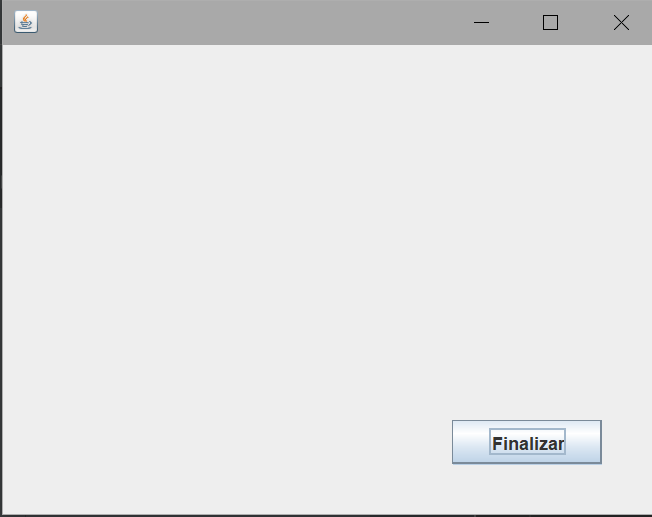
-El control **JButton** tiene como objetico que el usuario interactúe con él. La forma de usar la clase **JButton** se hace mediante la implementación de un ActionListener, más la declaración asociado a ese evento, en este caso el **actionPerformed**

#### Ejemplo 1 (Ventana con botón)

-Confeccionar una ventana que muestre un botón y que cuando se presione el botón (finalizar), se finaliza la ejecución del programa

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;   
  
  
public class Ejempl1 extends JFrame implements ActionListener {  
  
 JButton boton1;  
  
 public Ejempl1(){  
  
 //utilizo layout absoluto  
 setLayout(null);//  
 //Inicializo el boton  
 boton1 = new JButton("Finalizar");  
 //posiciono el boton al JFRame(ventana)  
 boton1.setBounds(300,250,100,30);  
 //añado el boton al JFrame  
 add(boton1);  
 //Añado el botón al Action Listener de esta clase  
 boton1.addActionListener(this);  
 }  
  
 public static void main (String[] args){  
 Ejempl1 ejemplo1 = new Ejempl1();  
 //Posiciono el JFrame en la propia pantalla pantalla  
 ejemplo1.setBounds(0,0,450,350);  
 //Muestro el JFrame  
 ejemplo1.setVisible(true);  
  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == boton1){  
 //salimos de la aplicacion  
 System.*exit*(0);  
 }  
 }

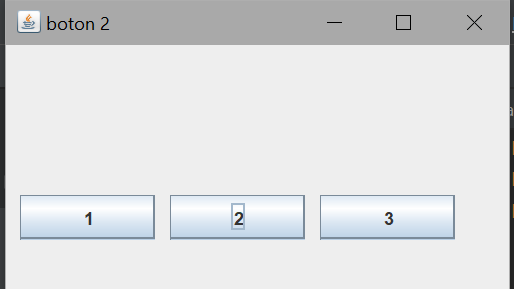
}



#### Ejemplo 2 (botones que cambien el titulo)

-Confeccionar una ventana que contenga tres objetos de la clase JButton con las etiquetas "1", "2" y "3". Al presionarse cambiar el título del JFrame indicando qué botón se presionó.

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
  
public class Ejemplo2 extends JFrame implements ActionListener {  
  
 private JButton boton1, boton2, boton3;  
  
 public Ejemplo2(){  
  
 setLayout(null);  
  
 //inicializo  
 boton1= new JButton("1");  
 boton2= new JButton("2");  
 boton3= new JButton("3");  
  
 //Posiciono  
 boton1.setBounds(10,100,90,30);  
 boton2.setBounds(110,100,90,30);  
 boton3.setBounds(210,100,90,30);  
  
 //Añado los botones al JFrame  
 add(boton1);  
 add(boton2);  
 add(boton3);  
  
 //Añado al Action Listener  
 boton1.addActionListener(this);  
 boton2.addActionListener(this);  
 boton3.addActionListener(this);  
  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 if (e.getSource() == boton1){  
 setTitle("boton 1");  
 }  
 else if (e.getSource() == boton2){  
 setTitle("boton 2");  
 }  
 else if (e.getSource() == boton3){  
 setTitle("boton 3");  
 }  
  
 }  
  
 public static void main(String[] args){  
 Ejemplo2 ejemplo2 = new Ejemplo2();  
 ejemplo2.setBounds(0,0,350,200);  
 ejemplo2.setVisible(true);  
 }  
}



#### Ejercicio 3 (Escribir o teclear números y limpiar)

-Confeccionar una ventana que contenga 3 objetos de la clase JButton con las etiquetas “Pulsa o teclea 1, Pulsa o teclea 2, Pulsa o teclea 3” y un display de tipo JLabel. Al presionar cualquiera de los botones o pulsar en el teclado el 1 o 2 o 3 debe cambiar el texto del JLabel con los números correspondientes.

-Además, al pasar por encima de cada boron, debe cambiar el color de fondo del mismo. Por último, un 4º botón con el texto “Limpiar display”(o pulsar la tecla c) que vaccie el texto del display.

public class Ejemplo3 extends JFrame {  
  
 JLabel display;//el label es una ventana  
 JButton boton1, boton2, boton3, botonlimpiar;  
  
 public Ejemplo3(){  
  
 //Iniciar display  
 initDisplay();  
 //Iniciar botones  
 initBotones();  
 //Iniciar acciones  
 initAciones();  
 //Iniciar los hover(ratoón pasa por encimade un elemento)  
 initHovers();  
 //Iniciar una pantalla (JFrame), se crea al final  
 initPantalla();  
  
 }  
  
 void initDisplay(){  
  
 //Propiedades y estilos del display  
 display = new JLabel("");  
 display.setBounds(15,15,195,60); //x ,y, width, height

display.setOpaque(true); //Hago el elemento opaco para poder darle un color de fondo  
 display.setBackground(Color.*BLACK*); //Color de fondo  
 display.setForeground(Color.*GREEN*); //Color de fuente  
 display.setBorder(new LineBorder(Color.*DARK\_GRAY*)); //Borde del display  
 display.setFont(new Font("MONOSPACED",Font.*PLAIN*,24)); //Tipo de fuente  
 display.setHorizontalAlignment(SwingConstants.*RIGHT*); //Alinear texto a la derecha  
 display.setFocusable(true); //Permito que el JLabel pueda tener el foco  
 add(display); //Añado el elemento al JFrame  
  
 //Eventos de entrada del teclado  
 display.addKeyListener(new KeyListener() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 String ch = String.*valueOf*(e.getKeyChar()); //Me devuelve el valor de tecla  
 if ("1".equals(ch) || "2".equals(ch) || "3".equals(ch)){  
 display.setText(display.getText() + ch); //Añado el texto al display  
 }  
 if ("c".equals(ch)){  
 display.setText(""); //Limpio el display  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
  
 }  
 });  
 }

void initBotones(){  
  
 boton1 = new JButton("Pulsa o tecla el 1");  
 boton2 = new JButton("Pulsa o tecla el 2");  
 boton3 = new JButton("Pulsa o tecla el 3");  
 botonlimpiar = new JButton("Limpiar display");

//les pongo las caracterisiticas  
 boton1.setBounds(15,90,195,50);  
 boton2.setBounds(15,155,195,50);  
 boton3.setBounds(15,220,195,50);  
 botonlimpiar .setBounds(15,285,195,50);

//Cambio la fuente  
 boton1.setFont(new Font("MONOSPACE", Font.*PLAIN*,16));  
 boton2.setFont(new Font("MONOSPACE", Font.*PLAIN*,16));  
 boton3.setFont(new Font("MONOSPACE", Font.*PLAIN*,16));  
 botonlimpiar.setFont(new Font("MONOSPACE", Font.*PLAIN*,16));

//les hago opacos  
 boton1.setOpaque(true);  
 boton2.setOpaque(true);  
 boton3.setOpaque(true);  
 botonlimpiar.setOpaque(true);  
  
 boton1.setFocusPainted(false);//le quito el marco azul que sale al hacer foco en el boton  
 boton1.setBackground(Color.*darkGray*); //cambiamos el color de fondo  
 boton1.setBorder(new LineBorder(Color.*darkGray*)); //cambiamos los bordes

boton1.setForeground(Color.*white*); //cambiamos el color de las letras  
  
 boton2.setFocusPainted(false);   
 boton2.setBackground(Color.*darkGray*);  
 boton2.setBorder(new LineBorder(Color.*darkGray*));  
 boton2.setForeground(Color.*white*);  
  
 boton3.setFocusPainted(false);  
 boton3.setBackground(Color.*darkGray*);  
 boton3.setBorder(new LineBorder(Color.*darkGray*));  
 boton3.setForeground(Color.*white*);  
  
 botonlimpiar.setFocusPainted(false);   
 botonlimpiar.setBackground(Color.*darkGray*);  
 botonlimpiar.setBorder(new LineBorder(Color.*darkGray*));  
 botonlimpiar.setForeground(Color.*white*);

//añado los botones las JFrame  
 add(boton1);  
 add(boton2);  
 add(boton3);  
 add(botonlimpiar);  
  
 }

//Creamos las accciones para que al pulsar la tecla se escriba dicho numero  
 void initAciones(){

boton1.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 //Recupero el exto del display y le añado un 1  
 display.setText(display.getText() + "1");

devuelvo el foco al display para poder utilizar las teclas  
 display.requestFocus();  
 }  
 });  
  
 boton2.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 //Recupero el exto del display y le añado un 2  
 display.setText(display.getText() + "2");  
 display.requestFocus();  
 }  
 });  
  
 boton3.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 //Recupero el exto del display y le añado un 3  
 display.setText(display.getText() + "3");  
 display.requestFocus();  
 }  
 });  
  
 botonlimpiar.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 //dejo vacio el display  
 display.setText("");  
 display.requestFocus();  
 }  
 });  
  
  
  
 }  
  
 void initHovers(){  
  
 //que se ponga verde cuando pase por encima del boton limpiar el raton  
 botonlimpiar.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 botonlimpiar.setBackground(Color.*green*);  
 botonlimpiar.setBorder(new LineBorder(Color.*green*));  
 botonlimpiar.setForeground(Color.*darkGray*);  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
 botonlimpiar.setBackground(Color.*darkGray*);  
 botonlimpiar.setBorder(new LineBorder(Color.*darkGray*));  
 botonlimpiar.setForeground(Color.*white*);  
  
 }  
 });  
  
 //que se ponga verde cuando pase por encima del boton1 el raton  
 boton1.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 boton1.setBackground(Color.*green*);  
 boton1.setBorder(new LineBorder(Color.*green*));  
 boton1.setForeground(Color.*darkGray*);  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
 boton1.setBackground(Color.*darkGray*);  
 boton1.setBorder(new LineBorder(Color.*darkGray*));  
 boton1.setForeground(Color.*white*);  
  
 }  
 });  
  
 //que se ponga verde cuando pase por encima del boton2 el raton  
 boton2.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 boton2.setBackground(Color.*green*);  
 boton2.setBorder(new LineBorder(Color.*green*));  
 boton2.setForeground(Color.*darkGray*);  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
 boton2.setBackground(Color.*darkGray*);  
 boton2.setBorder(new LineBorder(Color.*darkGray*));  
 boton2.setForeground(Color.*white*);  
  
 }  
 });  
  
 //que se ponga verde cuando pase por encima del boton3 el raton  
 boton3.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 boton3.setBackground(Color.*green*);  
 boton3.setBorder(new LineBorder(Color.*green*));  
 boton3.setForeground(Color.*darkGray*);  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
 boton3.setBackground(Color.*darkGray*);  
 boton3.setBorder(new LineBorder(Color.*darkGray*));  
 boton3.setForeground(Color.*white*);  
  
 }  
 });  
  
 }

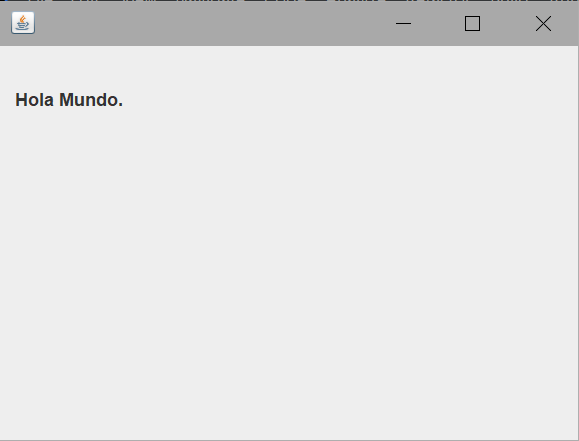
//Modificamos la ventana en sí  
 void initPantalla(){  
  
 setTitle("Ejemplo3");//Pongo el titulo a la ventana  
 setBounds(0,0,240,390);//Tamaño a la ventana y posicion en la pantalla  
 //setSize(240,275);//Tamaño de la ventana  
 setResizable(false);//Impido que la ventana se pueda remdimensionar  
 setLayout(null);//Utilizo Layout absoluto  
 getContentPane().setBackground(Color.*BLACK*);//Fondo de la ventana de color negrto  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);//Al cerrar la ventana, cieero el proceso  
 setVisible(true);//muestro la ventana  
  
 }  
  
 public static void main (String[] args){  
 new Ejemplo3();  
 }

}



#### Ejercicio 4 (Hola Mundo)

import javax.swing.\*;  
  
/\* La clase JFrame encapsula el concepto de una ventana,  
 para implementar una aplicación que muestre una ventana debemos plantear una clase que herede de la clase JFrame \*/  
public class Ejemplo4 extends JFrame{  
  
 /\* Definimos un objeto JLabel para la etiqueta. \*/  
 private JLabel label1;  
  
 /\* En el constructor de la clase llamamos al método heredado de la clase JFrame llamado setLayout  
 y le pasamos como parámetro un valor null, con esto estamos informándole a la clase JFrame que  
 utilizaremos posicionamiento absoluto para los controles visuales dentro del JFrame\*/  
 public Ejemplo4() {  
  
 setLayout(null);  
  
 /\* Creamos el objeto de la clase JLabel y le pasamos como parámetro al constructor el texto a mostrar.\*/  
 label1=new JLabel("Hola Mundo.");  
  
 /\* Ubicamos al objeto de la clase JLabel llamando al método setBounds,  
 este requiere como parámetros la columna, fila, ancho y alto del JLabel.  
 Finalmente llamamos al método add (metodo heredado de la clase JFrame)  
 que tiene como objetivo añadir el control JLabel al control JFrame \*/  
 label1.setBounds(10,20,200,30);  
 add(label1);  
 }  
  
 /\* En el método main creamos un objeto de la clase Formulario, llamamos al  
 método setBounds para ubicar y dar tamaño al control y mediante el método setVisible hacemos visible el JFrame \*/  
 public static void main(String[] args) {  
  
 Ejemplo4 formulario1=new Ejemplo4();  
 formulario1.setBounds(10,10,400,300);  
 formulario1.setVisible(true);  
 }  
  
}



#### Ejercicio5 (Ventana Principal y Secundaria)

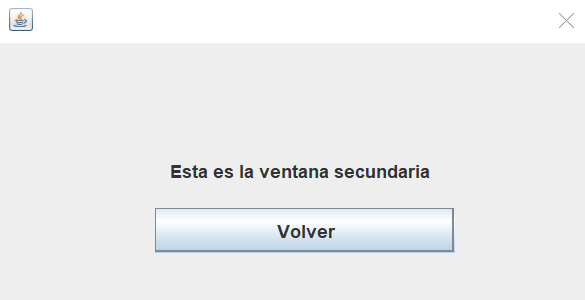
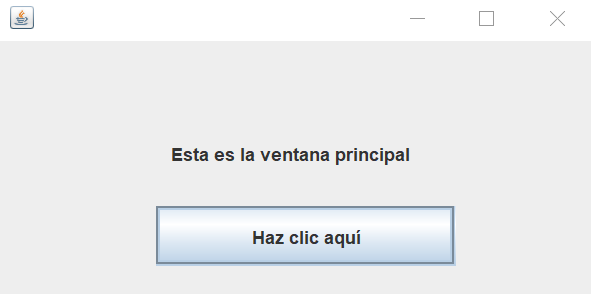
-Necesitamos crear dos clases, una para cada ventana, siendo una de ellas la principal donde al pulsar “Haz clic aquí” aparezca otra ventana con el botón “volver” y volver a la otra ventana

VENTANA PRINCIPAL

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Ejemplo5Principal extends JFrame implements ActionListener {  
  
 private JButton boton;  
 private JLabel label;  
  
 public Ejemplo5Principal(){  
  
 //JFrame  
 setLayout(null);  
 setBounds(10,10,410,210);  
  
 //Etiqueta  
 label = new JLabel("Esta es la ventana principal");  
 label.setBounds(115,60,200,30);  
 add(label);  
  
 //Botón  
 boton = new JButton("Haz clic aquí");  
 boton.setBounds(105,110,200,40);  
 add(boton);  
  
 //Añado el botón al action listener  
 boton.addActionListener(this);  
  
 //Muestro el JFrame  
 setVisible(true);  
  
 }  
  
 public static void main(String[] args) {  
 new Ejemplo5Principal();  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == boton){  
 //abrir la ventana secundaria  
 //poner true en el modal hace que solo deja abrir una ventana secundaria  
 Ejemplo5Secundaria secundaria = new Ejemplo5Secundaria(this,true);  
 secundaria.setVisible(true);//hago que la ventana sea visible  
  
 }  
 }  
}

VENTANA SECUNDARIA

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Ejemplo5Secundaria extends JDialog implements ActionListener {  
  
  
 private JButton boton2;  
 private JLabel label1;  
  
 public Ejemplo5Secundaria(Ejemplo5Principal parent, boolean modal){//parent nos llama al padre(osea a la ventana principal)  
  
 //Invocar al constructor de la clase superior  
 super(parent, modal);  
  
 setLayout(null);  
 setBounds(440,10,410,210);  
  
 label1 = new JLabel("Esta es la ventana secundaria");  
 label1.setBounds(115,60,500,50);  
 add(label1);  
  
 boton2 = new JButton("Volver");  
 boton2.setBounds(105,110,200,30);  
 add(boton2);  
  
 boton2.addActionListener(this);  
  
  
  
 }  
  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == boton2){  
 //Cierro la ventana  
 setVisible(false);  
 }  
 }  
}



### Práctica 1: Pares o Nones

1º Tú serás pares y debes poder elegir del 1 al 5 en cada tirada

2º La máquina será nones y elegirá un número del 1 al 5 en cada tirada

3º Un texto debe mostrar, en otra ventana, el número que tú elegiste, el que eligió la máquina y el ganador

4ºEsta última ventana permitirá volver a la ventana anterior.

### Práctica 2: Diana Horizontal

-Desarrollar un juego cuyo objetivo es acertar en el medio de un control de tipo deslizante (JSlider) con las siguientes especificaciones:

1ºEl control debe tener un ancho aprox del 50% de la pantalla

2º Al cambiar el valor del control, otra pantalla debe devolver el valor conseguido entre 0 y 100, asi como un botón que permita volver a empezar.

3ºModifica ahora el valor devuelto con las siguientes especificaciones:

* Si es 50 🡪 texto en rojo y decir ¡Enhorabuena, acertaste¡
* Si es entre 40 y 60 🡪 sexto en naranja y decir “Ha estado cerca, prueba otra vez”
* En cualquier otro caso, debe mostrar el texto verde

4ºAñade un selector que permita escoger un nivel de juego entre 1 y 3 . Al seleccionar un nivel, el ancho del deslizador deber, aprox de 10%, 50%(defecto) y 100%

5ºAñade un botón arriba a la derecha con el texto “Acerca de”. Al pulsarlo se debe abrir una ventana con tu nombre y una descripción de los elementos utilizados (JSlider, JButton,etc) y su funcionamiento)