

Clase 12: Ventanas

Problema: mostrar la palabra “hola” en una ventana



```
import java.awt.*;
class Hola{
    static public void main(String[]args){
        Ventana v = new Ventana();
        v.show(); //muestra ventana en pantalla
        //v.setVisible(true); si no funciona v.show()
    }
}
class Ventana extends Frame{
    private Label palabra = new Label("hola");
    public Ventana(){
        setSize(50,50);
        add(palabra);
    }
}
```

Explicaciones

1. **import java.awt.*;**
deja disponibles (visibles) clases predefinidas para crear ventanas (Frame) y sus componentes (Label, TextField, Button, Panel)
2. **class Ventana extends Frame{...}**
hereda componentes de clase Frame
3. **private Label palabra = new Label(...);**
Label: clase predefinida para componentes que permiten mostrar texto
4. **public Ventana() {...}**
constructor que inicializa objeto de clase Ventana
5. **setSize(50,50);**
método, de la clase Frame o antecesoras, que define ancho (width) y alto (height) de la ventana en pixeles (picture elements: puntos gráficos)
6. **add(palabra);**
agrega componente (Label palabra) al objeto que representa la ventana

Solución 2: una sólo clase con método main y constructor

```
import java.awt.*;
class Hola extends Frame
{
    static public void main(String[]args){
        //crear objeto e invocar método
        new Hola().show();
    }
    public Hola(){
        setSize(50,50);
        add(new Label("hola"));
    }
}
```

Problema. Programar el sgte saludador personalizado ☺

nombre	ventana	clase	objetivo
pregunta	Cuál es tu nombre?	Label	mostrar texto
nombre	Juan	TextField	ingresar texto
respuesta	Hola Juan	Label	mostrar texto
quit	quit	Button	aceptar click

```
import java.awt.*; import java.awt.event.*;
class Hola extends Frame
{
    static public void main(String[]args){new Hola().show();}

    //representación (componentes del objeto)
    private Label
        pregunta=new Label("Cuál es tu nombre?"),
        respuesta=new Label("");
    private TextField nombre=new TextField();
    private Button quit=new Button("quit");
}
```

```
//constructor
public Hola()
{
    //diagramar ventana
    setLayout(new GridLayout(4,1)); //4 filas, 1 col
    setSize(200,50*4); //ancho:200, alto:200

    //agregar componentes
    add(pregunta); //fila 1, col 1
    add(nombre); //fila 2, col 1
    add(respuesta); //fila 3, col 1
    add(quit); //fila 4, col 1

    //crear y activar objetos para atender eventos
    //escuchador para TextField nombre
    nombre.addActionListener(new EscuchadorNombre());
    //escuchador para Button quit
    quit.addActionListener(new EscuchadorQuit());
}
```

```
class EscuchadorNombre implements ActionListener
{
    public void actionPerformed(ActionEvent x){
        String s=nombre.getText();
        respuesta.setText("hola "+s);
    }
}
class EscuchadorQuit implements ActionListener
{
    public void actionPerformed(ActionEvent x){
        System.exit(0);
    }
}
```

Clase 12: Ventanas

Explicaciones

1. `import java.awt.event.*;`
 - incluye clases e interfaces asociadas con eventos producidos por componentes
2. `private TextField nombre=new TextField();`
 - define componente para ingresar texto
 - TextField: clase predefinida con métodos `setText` y `getText` (igual que Label)
 - genera evento al presionar tecla enter (return)
3. `private Button quit=new Button("quit");`
 - define componente para aceptar click del mouse
 - Button: clase predefinida con métodos `setLabel` y `getLabel`
 - genera evento con click del mouse sobre el botón

4. `setLayout(new GridLayout(4,1));`

- método que define diagramación de componentes de ventana
- `new GridLayout(f, c)`: objeto que diagrama ventana de `f` filas y `c` columnas (todas las componentes de igual tamaño)

5. `nombre.addActionListener(new EscuchadorNombre());`

- crea y agrega (activa) objeto que “escuchará” nombre, es decir, que “despertará” y tomará el control al ingresar tecla enter
- `addActionListener`: método de clase TextField (o superiores)

6. `quit.addActionListener(new EscuchadorQuit());`

- crea y activa objeto que “atenderá” click del mouse

7. `class EscuchadorNombre implements ActionListener{...}`

- clase encajonada permite acceder a componentes privadas
- implementación de:

```
interface ActionListener{
    public void actionPerformed(ActionEvent x);
}
```
- método `actionPerformed`
 - recibe el control al presionarse tecla enter en TextField nombre
 - obtiene nombre y muestra respuesta
 - vuelve a quedar esperando evento (enter) en nombre

8. `class EscuchadorQuit implements ActionListener{...}`

- método `actionPerformed`
 - recibe el control al clickear el botón quit
 - termina (aborta) el programa

Flujo de ejecución

1. método main crea objeto de clase Hola
2. constructor diagrama ventana, agrega componentes y activa “escuchadores”
3. método main invoca método `show` (muestra ventana) y termina
4. objetos “escuchadores” esperan que se produzca un evento
5. si se ingresa enter en TextField nombre, objeto escuchador obtiene nombre, muestra respuesta y regresa a esperar un nuevo nombre
6. si se da click en botón quit, objeto escuchador termina el programa

Solución 2. Con un sólo objeto escuchador

```
class Hola extends Frame{
    static public void main(String[] args){new Hola().show();}
    private ...
    public Hola(){
        ...
        Escuchador escuchador = new Escuchador();
        nombre.addActionListener(escuchador);
        quit.addActionListener(escuchador);
    }
    class Escuchador implements ActionListener{
        public void actionPerformed(ActionEvent x){
            if(x.getSource()==nombre) //evento en nombre?
                respuesta.setText("hola "+nombre.getText());
            else //evento en botón quit?
                System.exit(0);
        }
    }
}
Nota. x.getSource() devuelve ref a componente que causó evento
```

Solución 3. Con una sólo clase con componentes y todos los métodos

```
class Hola extends Frame implements ActionListener{
    static public void main(String[] args){new Hola().show();}
    private ...
    public Hola(){
        ...
        //escuchador será este mismo objeto (de ref this)
        //creado en método main con new Hola()
        nombre.addActionListener(this);
        quit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent x){
        if(x.getSource()==quit) System.exit(0);
        respuesta.setText("hola "+nombre.getText());
    }
}
```

Clase 12: Ventanas

Problema. Diálogo para adivinar un número al azar entre 1 y 100:

Nombre	Contenido	Clase
pregunta	X(1-100)?	Label
numero	23	TextField
respuesta	X>23 o X<23 o X=23	Label
jugar	jugar de nuevo	Button
quit	quit	Button

Nota. Jugar de nuevo genera un nuevo X y “limpia” numero y respuesta

```
class Juego extends Frame implements ActionListener{
static public void main(String[]a){new Juego().show();}
private int X=U.azar(1,100);
private Label
    pregunta=new Label("X(1-100)?"),
    respuesta=new Label("");
private TextField numero=new TextField();
private Button
    quit=new Button("quit"),
    jugar=new Button("jugar de nuevo");
```

```
//constructor
public Juego()
{
    setLayout(new GridLayout(5,1));
    setSize(200,50*5);

    add(pregunta);
    add(numero);
    add(respuesta);
    add(jugar);
    add(quit);

    numero.addActionListener(this);
    jugar.addActionListener(this);
    quit.addActionListener(this);
}
```

```
public void actionPerformed(ActionEvent x)
{
    if(x.getSource()==quit) System.exit(0);

    if(x.getSource()==numero)
    {
        int n=Integer.parseInt(numero.getText());
        if(X<n)
            respuesta.setText("X<"+n);
        else if(X>n)
            respuesta.setText("X>"+n);
        else
            respuesta.setText("X="+n);
    }
    else if(x.getSource()==jugar)
    {
        X=U.azar(1,100);
        numero.setText("");
        respuesta.setText("");
    }
}
```

Modificar la interfaz a:

	Label	TextField	
p1	X(1-100)?	23	Panel
respuesta	X<23		Label
P2	jugar de nuevo	quit	Panel
	Button	Button	

```
private Label
    pregunta=new Label("X(1-100)?"),
    respuesta = new Label("");
private TextField
    numero = new TextField();
private Button
    jugar = new Button("jugar de nuevo"),
    quit = new Button("quit");
```

```
public Juego()
{
    //crear y diagramar panel p1
    Panel p1=new Panel();
    p1.setLayout( new GridLayout(1,2) );
    p1.add(pregunta);
    p1.add(numero);

    //crear y diagramar panel p2
    Panel p2=new Panel();
    p2.setLayout( new GridLayout(1,2) );
    p2.add(jugar);
    p2.add(quit);
}
```

```
//diagramar ventana
setLayout(new GridLayout(3,1));
setSize(200,50*3);

//agregar componentes a ventana
add(p1);
add(respuesta);
add(p2);

//activar escuchador
numero.addActionListener(this);
jugar.addActionListener(this);
quit.addActionListener(this);
}
```

Propuesto. Agregar botones “ayuda” y “rendirse”