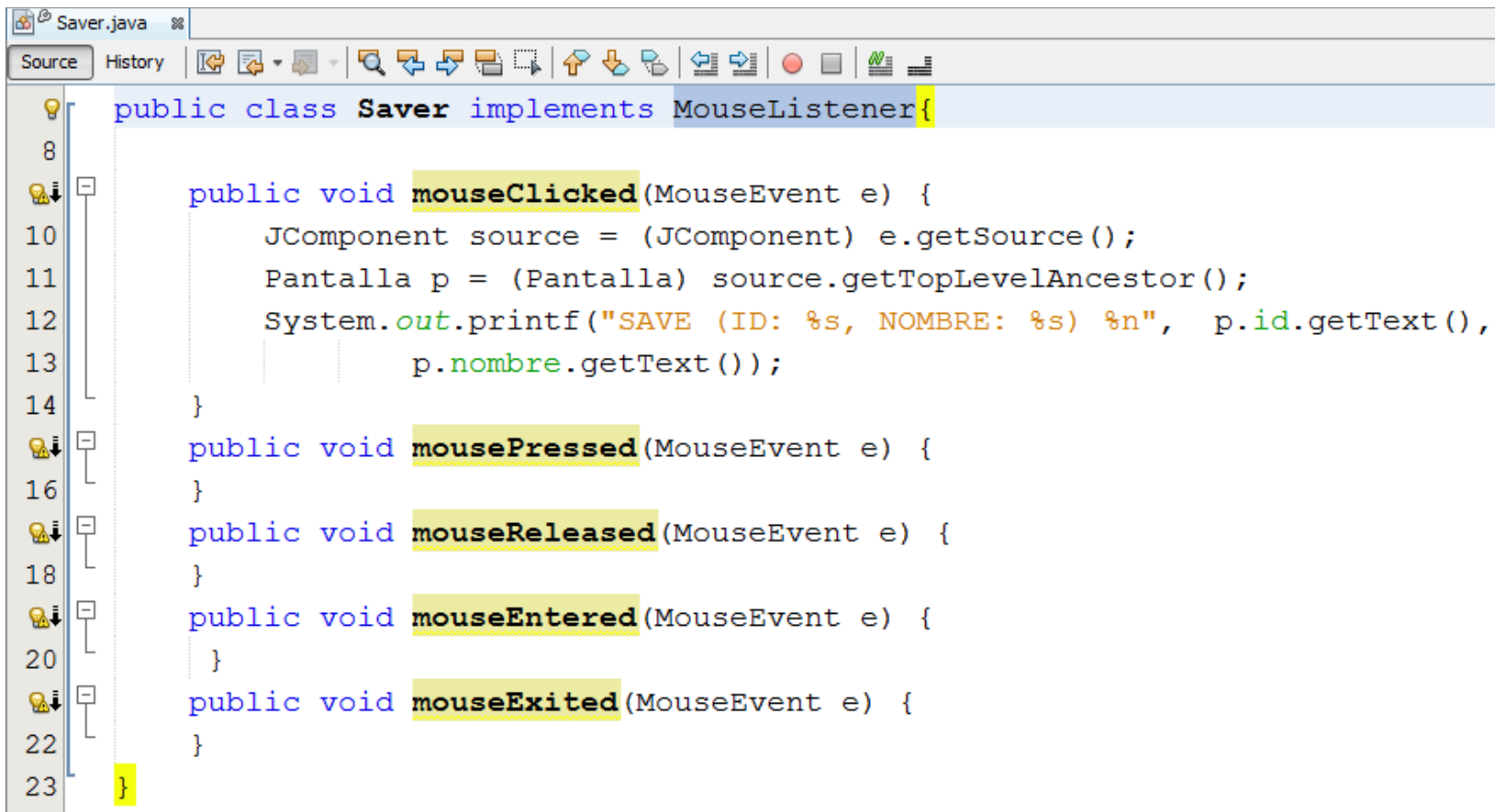


# Ejemplo: Saver

- Una clase Saver que “salve” los datos de una pantalla cuando se haga *click* sobre un botón.
- Implementa la interface *MouseListener*.
- Solo el caso *mouseClicked* es relevante.



```
public class Saver implements MouseListener {

    public void mouseClicked(MouseEvent e) {
        JComponent source = (JComponent) e.getSource();
        Pantalla p = (Pantalla) source.getTopLevelAncestor();
        System.out.printf("SAVE (ID: %s, NOMBRE: %s) %n", p.id.getText(),
            p.nombre.getText());
    }

    public void mousePressed(MouseEvent e) {
    }

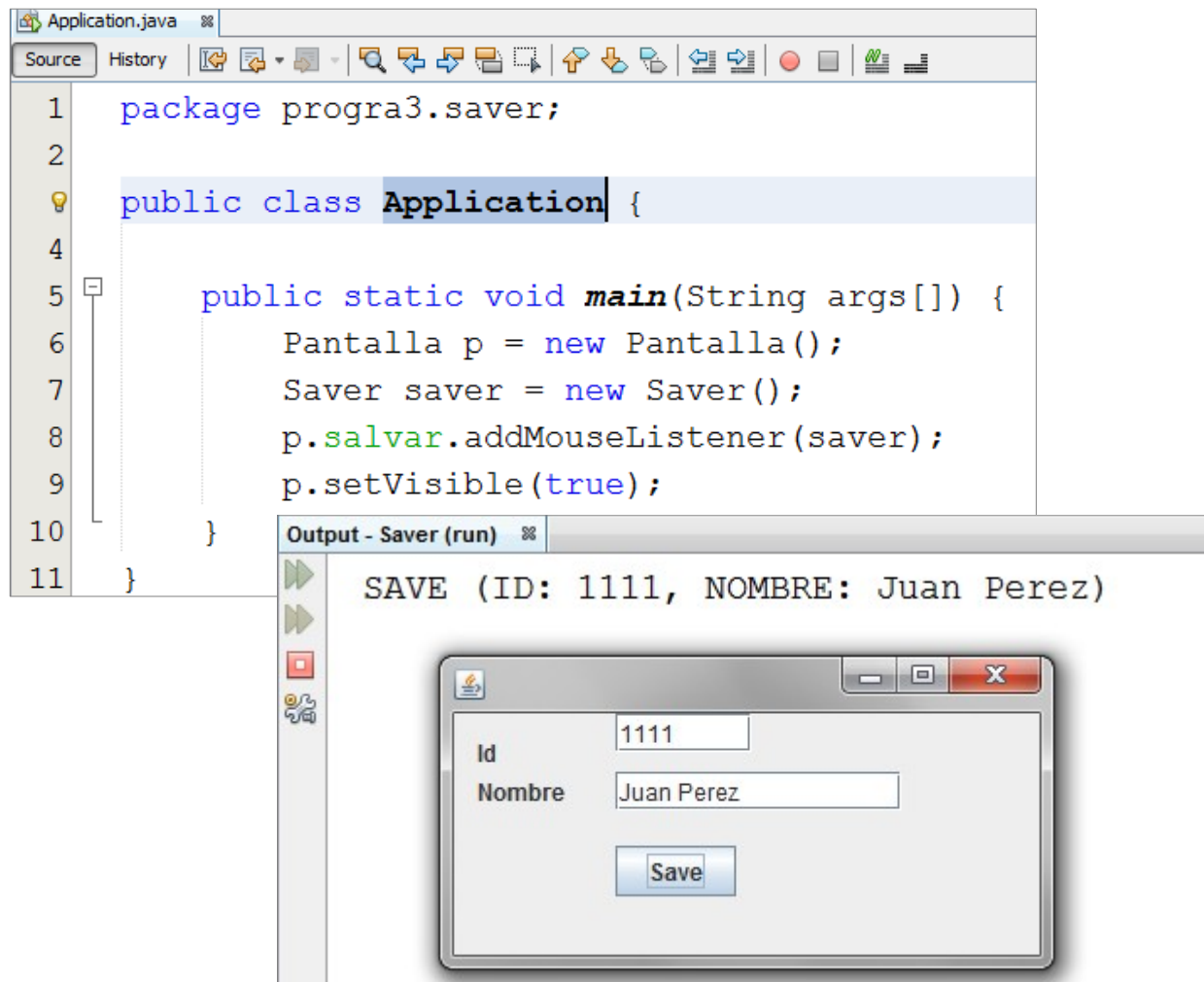
    public void mouseReleased(MouseEvent e) {
    }

    public void mouseEntered(MouseEvent e) {
    }

    public void mouseExited(MouseEvent e) {
    }
}
```

# Ejemplo: Saver ..

- Un objeto (*saver*) se registra como un *MouseListener* de un componente  *JButton* de una pantalla (*p.salvar*)



The screenshot shows an IDE window titled 'Application.java'. The code defines a package 'progra3.saver' and a public class 'Application'. Inside the class, there is a 'main' method that creates a 'Pantalla' object 'p', a 'Saver' object 'saver', registers 'saver' as a 'MouseListener' for 'p.salvar', and sets the window to be visible. The output window below the code shows the message 'SAVE (ID: 1111, NOMBRE: Juan Perez)'. A small window titled 'Save' is also shown, containing two text fields: 'Id' with the value '1111' and 'Nombre' with the value 'Juan Perez', and a 'Save' button.

```
1 package progra3.saver;
2
3 public class Application {
4
5     public static void main(String args[]) {
6         Pantalla p = new Pantalla();
7         Saver saver = new Saver();
8         p.salvar.addMouseListener(saver);
9         p.setVisible(true);
10    }
11 }
```

Output - Saver (run) %

SAVE (ID: 1111, NOMBRE: Juan Perez)

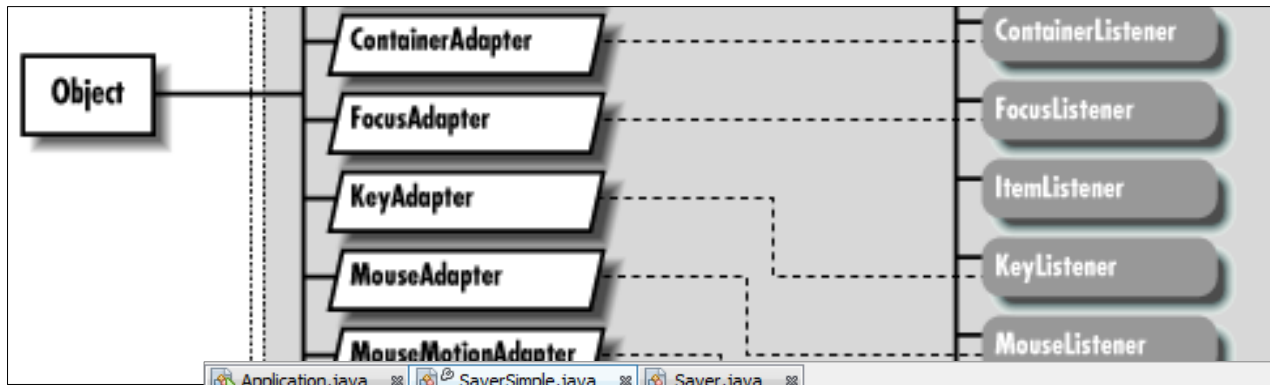
Save

Id 1111

Nombre Juan Perez

# Adaptadores

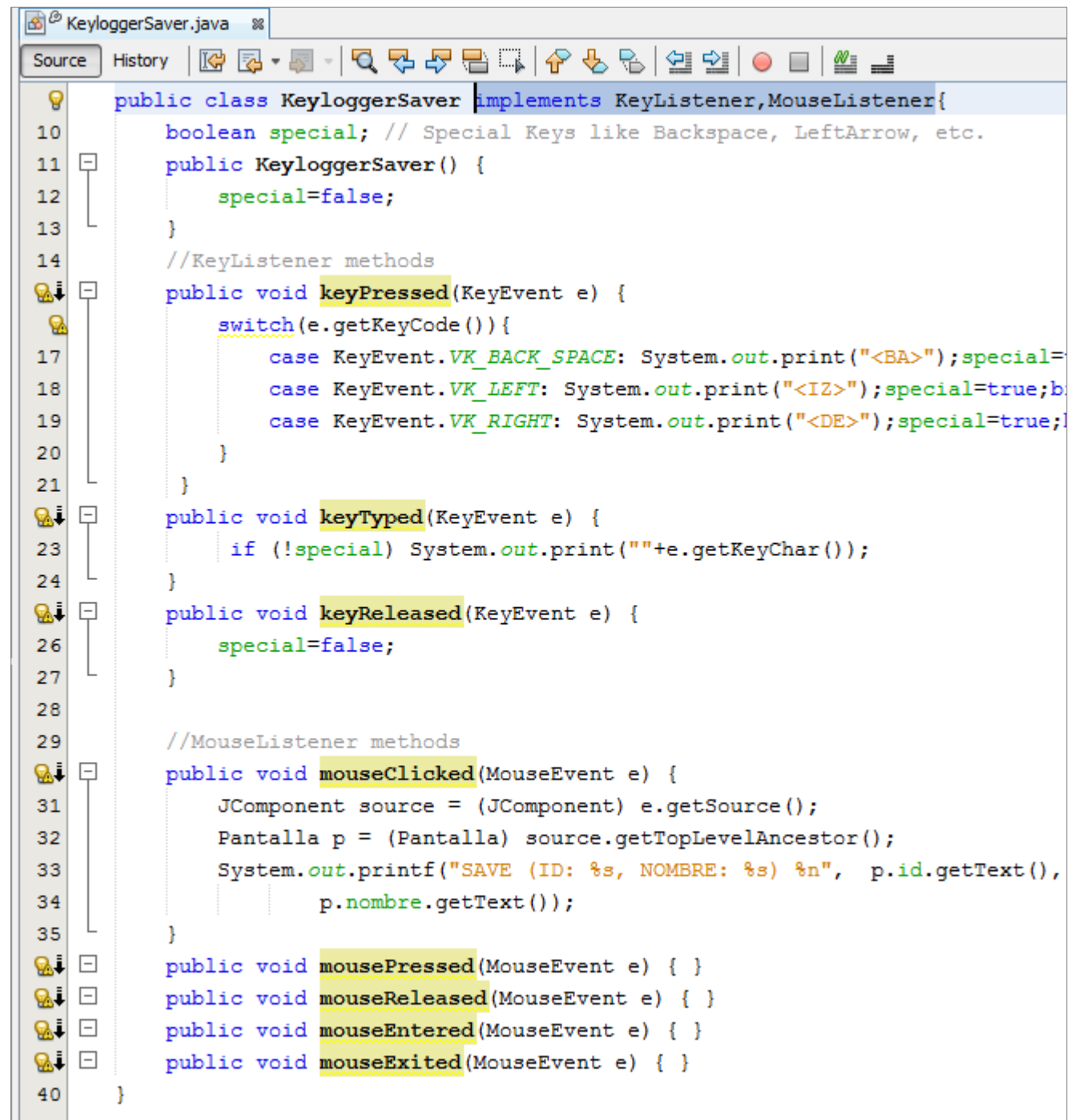
- Para cumplir una interface *listener* hay que proveer todos sus métodos, aunque solo algunos sean relevantes
- Las clases *Adapter* proveen implementación comodín
- Se puede heredar y sobre-escribir los métodos relevantes



```
Application.java  SaverSimple.java  Saver.java
Source  History  [Icons]
10  public class SaverSimple extends MouseAdapter implements MouseListener {
11
12      public void mouseClicked(MouseEvent e) {
13          JComponent source = (JComponent) e.getSource();
14          Pantalla p = (Pantalla) source.getTopLevelAncestor();
15          System.out.printf("SAVE (ID: %s, NOMBRE: %s) %n", p.id.getText(),
16                          p.nombre.getText());
17      }
18  }
```

# Múltiples *Listeners*

- Una misma clase puede implementar varias interfaces y así atender diversos tipos de eventos



```
KeyloggerSaver.java
Source History
public class KeyloggerSaver implements KeyListener,MouseListener{
    boolean special; // Special Keys like Backspace, LeftArrow, etc.
    public KeyloggerSaver() {
        special=false;
    }
    //KeyListener methods
    public void keyPressed(KeyEvent e) {
        switch(e.getKeyCode()){
            case KeyEvent.VK_BACK_SPACE: System.out.print("<BA>");special=
            case KeyEvent.VK_LEFT: System.out.print("<IZ>");special=true;b
            case KeyEvent.VK_RIGHT: System.out.print("<DE>");special=true;l
        }
    }
    public void keyTyped(KeyEvent e) {
        if (!special) System.out.print(""+e.getKeyChar());
    }
    public void keyReleased(KeyEvent e) {
        special=false;
    }
    //MouseListener methods
    public void mouseClicked(MouseEvent e) {
        JComponent source = (JComponent) e.getSource();
        Pantalla p = (Pantalla) source.getTopLevelAncestor();
        System.out.printf("SAVE (ID: %s, NOMBRE: %s) %n", p.id.getText(),
            p.nombre.getText());
    }
    public void mousePressed(MouseEvent e) { }
    public void mouseReleased(MouseEvent e) { }
    public void mouseEntered(MouseEvent e) { }
    public void mouseExited(MouseEvent e) { }
}
```

# Múltiples *Listeners* ...

- Un objeto (*loggerSaver*) se registra como *KeyListener* de un *TextField* (*p.nombre*) y como *MouseListener* de *JBButton* (*p.salvar*)

The screenshot displays an IDE with two main panels. The top panel, titled 'Application.java', shows the following Java code:

```
1 package progra3.keyloggersaver;
2
3 public class Application {
4
5     public static void main(String args[]) {
6         Pantalla p = new Pantalla();
7         KeyloggerSaver loggersaver = new KeyloggerSaver();
8         p.nombre.addKeyListener(loggersaver);
9         p.salvar.addMouseListener(loggersaver);
10        p.setVisible(true);
11    }
12 }
```

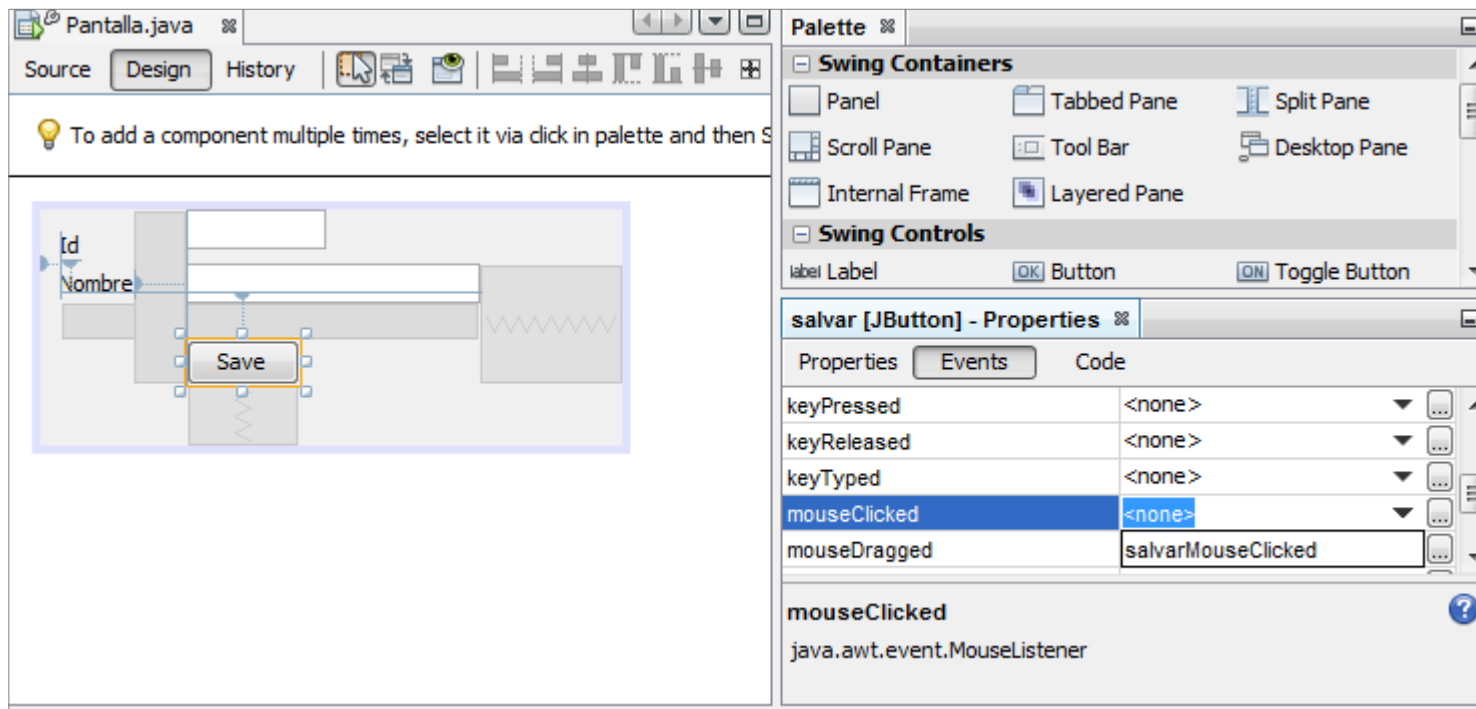
The bottom panel, titled 'Output - KeyLoggerSaver (run)', shows the output of the program:

```
JOAN<IZ><IZ><BA>U<DE><DE> PEREZ SAVE (ID: 111, NOMBRE: JUAN PEREZ )
```

Below the output, a small window titled 'Application' is shown. It contains two text input fields: 'Id' with the value '111' and 'Nombre' with the value 'JUAN PEREZ'. Below these fields is a button labeled 'Save'.

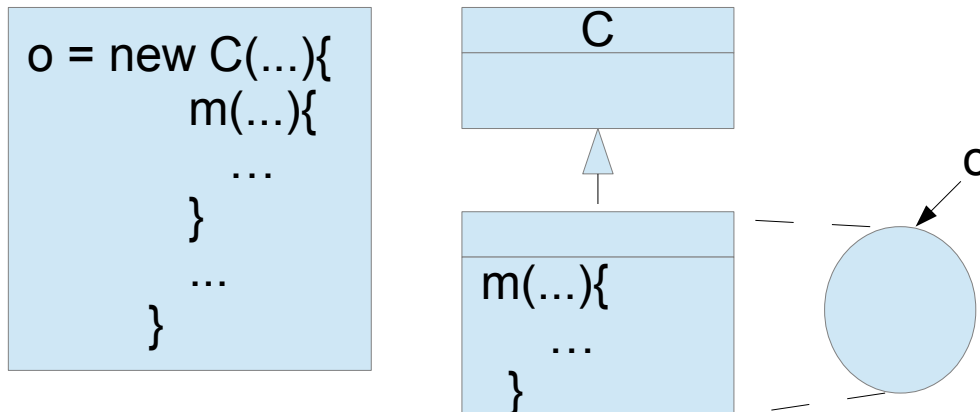
# Listeners con clases internas anónimas

- El editor de pantallas de NetBeans implementa los *handlers* para los eventos como métodos de la propia pantalla.
- Pero intermediados por *Listeners* implementados como clases internas anónimas



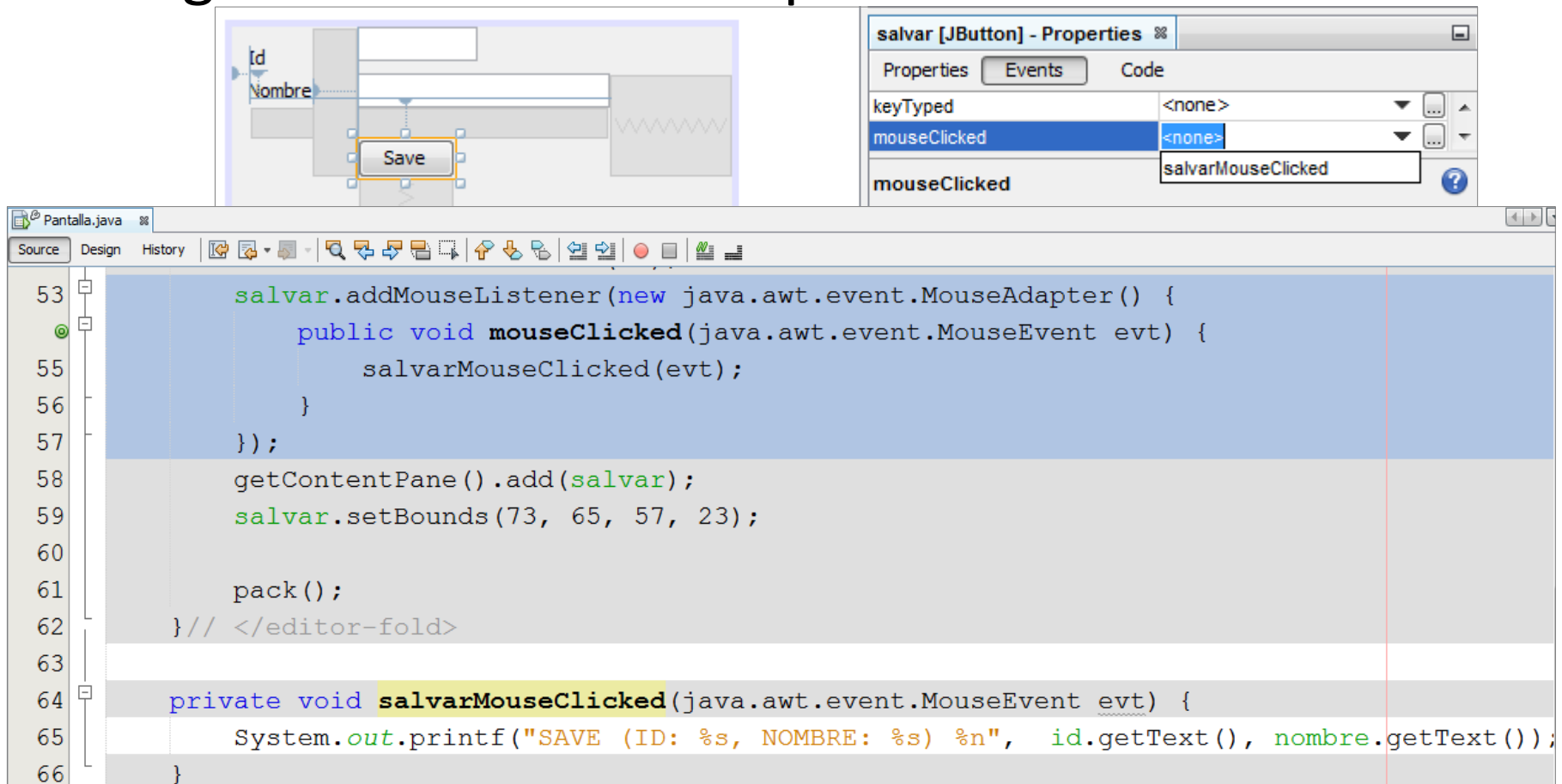
# Clases internas anónimas

- Una clase interna es una que está definida dentro de otra clase, y tiene acceso a los elementos de ella.
- Una clase anónima no tiene nombre, solo se usa para contruir una instancia.
- Se genera al crear una instancia de una clase o interface existente y sobre-escribir o agregar algunos métodos



# Listeners con clases internas anónimas

- NetBeans crea una instancia de una clase anónima que hereda de un *Adapter*, y por tanto implementa *Listener*.
- Sobre-escribe el método correspondiente, el cual delega a un método de la pantalla.





# Ciclo de Atención de Eventos

- El hilo principal (*main*) crea la interface (pantalla) y continua su propia ejecución hasta terminar.
- Un hilo aparte, el EDT (Event Dispatcher Thread) procesa los eventos de la interface de usuario (mouse, teclado, etc.)

