

Interacción Persona Computador

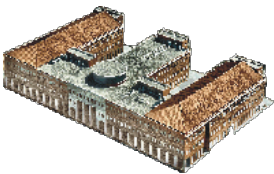
Laboratorio – S04

The logo of the University of Valencia (UVa) is a red square with the white text "UVa" inside.

UVa

Aplicación del patrón MVC al diseño de GUI

Y. Crespo/M.Gonzalo/C.Hernández/A. Martínez



EI-INFORMATICA

Introducción

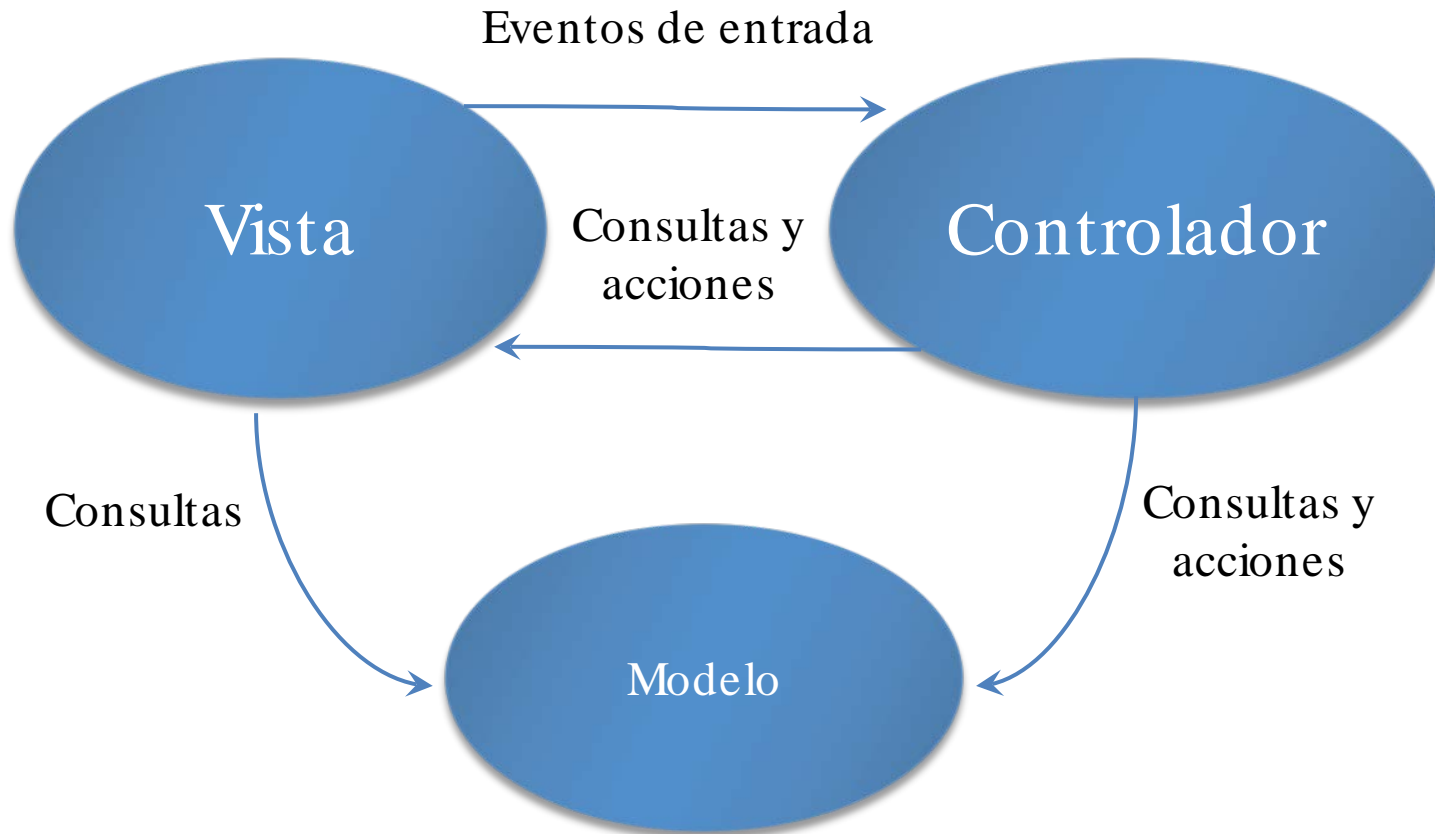
- Hoy veremos
 - Introducción al patrón arquitectónico MVC
 - Cómo implementar el patrón MVC en una aplicación Java
 - Práctica a partir de la primera aplicación creada con NetBeans

Patrón MVC

- ❑ Patrón de arquitectura de software
- ❑ Desarrollado para Smalltalk-80 (Xerox - PARC)
- ❑ Vista: maneja la salida
 - ❑ Solo muestra información: "frame"
 - ❑ Escucha los cambios de modelo y actualiza sus componentes
- ❑ Controlador de esa vista: entrada
 - ❑ Escucha los eventos de entrada de la vista
 - ❑ Manipula el modelo y provoca que la vista se actualice
- ❑ Modelo: información acerca del dominio
 - ❑ Implementa el cambio de estado
 - ❑ Puede ser un objeto del modelo de dominio

Patrones de diseño software para GUI

Patrón MVC



NOTA: Las dependencias mostradas en el esquema no se traducen necesariamente en relaciones directas entre objetos

Patrón MVC

- Dos versiones del patrón

- MVC activo:

- El modelo notifica de sus cambios a la vista

- MVC pasivo:

- El modelo no tiene acceso a la vista ni al controlador.
 - Si sufre un cambio de estado, puede no reflejarse automáticamente en la interfaz.

Patrón MVC

□ Flujo de control

- El usuario realiza una acción en la interfaz(vista)
- El controlador de la vista trata el evento de entrada
 - Previamente se ha registrado
- El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta)
- (Opcionalmente) El controlador provoca que la vista se actualice. La vista toma los datos del modelo
 - El modelo no tiene conocimiento directo de la vista
- La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo

MVC en JavaSwing

- **Modelo:**

- El modelo lo realiza el desarrollador

- **Vista:**

- Conjunto de objetos de clases que heredan de `java.awt.Component`

- **Controlador:**

- El controlador es el thread de tratamiento de eventos, que captura y propaga los eventos a la vista y al modelo
 - Clases de tratamiento de los eventos (clases internas y anónimas) que implementan interfaces de tipo `EventListener` (`ActionListener`, `MouseListener`, `WindowListener`, etc.)

Patrón MVC

- Separación de responsabilidades
 - Cada módulo es responsable de un aspecto
- Ventajas: Desacoplamiento
 - El modelo y la vista están desacoplados, pueden cambiar de forma independiente
 - El modelo puede ser usado con otras vistas
 - Varias vistas pueden compartir simultáneamente un modelo
 - Las vistas se pueden reutilizar para otros modelos, siempre que el modelo implemente la interfaz

Patrón MVC

□ Vista

- Necesita “ver” al Controlador y al Modelo (*private*)
- En el constructor de la Vista, se crea el Modelo y el Controlador

```
public class VistaNA extends javax.swing.JFrame {  
    private ControladorNA miControl;  
    private ModeloNA miModelo;  
  
    public VistaNA() {  
        initComponents();  
        miModelo = new ModeloNA();  
        miControl = new ControladorNA(this,miModelo);  
    }  
}
```

Patrón MVC

□ Controlador

- Necesita “ver” a la Vista y al Modelo (*private*)
- El constructor del Controlador tiene como parámetros la Vista y el Modelo

```
public class ControladorNA {  
    private VistaNA miVista;  
    private ModeloNA miModelo;  
  
    public ControladorNA(VistaNA v, ModeloNA m){  
        miVista = v;  
        miModelo = m;  
    }  
}
```

Patrón MVC

□ Modelo

- El Modelo sólo necesita conocer su información (*private*)

```
public class ModeloNA {  
    float resultado;  
  
    public ModeloNA() {  
        ...  
    }  
}
```

Ejemplo

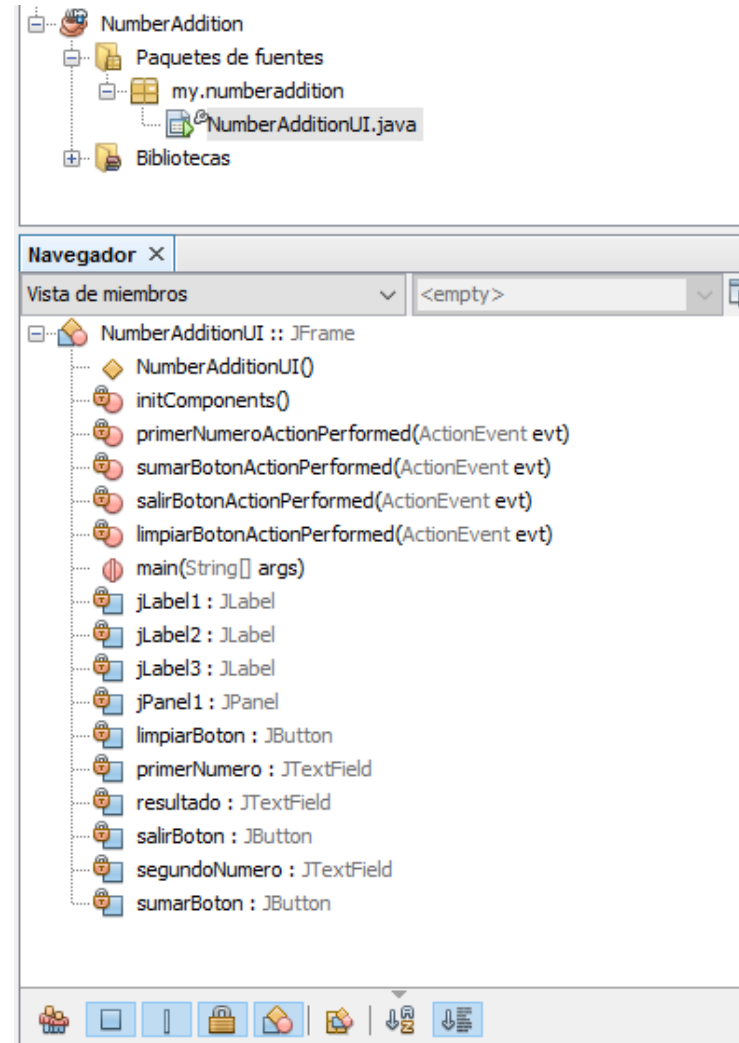
Conversión de la aplicación NumberAddition a arquitectura
MVC

Ejemplo - NumberAddition

□ Observamos el código generado en la aplicación NumberAddition

□ Todas las responsabilidades son asumidas por la misma clase (M, V y C, a la vez)

NumberAdditionUI



Ejemplo - NumberAddition

- Los eventos se manejan mediante la creación de una clase interna anónima que implementa el método **actionPerformed** declarado en “ActionListener”, y que llama a un manejador que se hace cargo de la tarea asociada al evento.

```
sumarBoton.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        sumarBotonActionPerformed(evt);  
    }  
});
```

- *Este código es generado automáticamente por NetBeans, de manera que el programador sólo se tiene que hacer cargo de codificar lo que tiene que hacer el manejador que se hace cargo de la tarea asociada al evento.*

Ejemplo - NumberAddition

```
sumarBoton.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        sumarBotonActionPerformed(evt);  
    }  
});
```

```
private void sumarBotonActionPerformed(java.awt.event.ActionEvent evt) {  
    float num1, num2, result;  
    num1 = Float.parseFloat(primerNumero.getText());  
    num2 = Float.parseFloat(segundoNumero.getText());  
    result = num1 + num2;  
    resultado.setText(String.valueOf(result));  
}
```

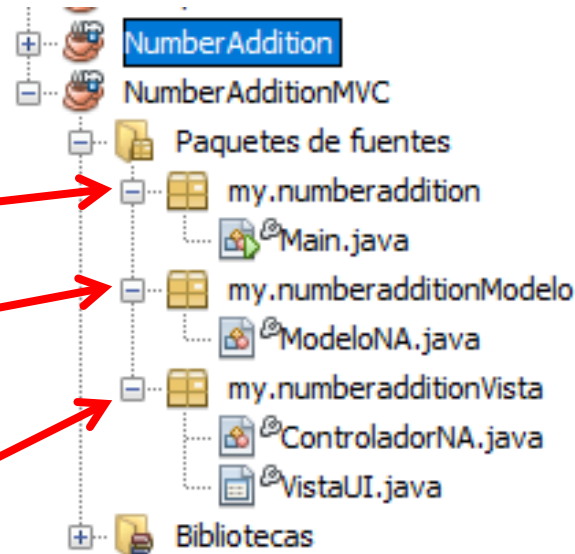
Ejemplo - NumberAddition

- ¿Cómo convertir NumberAddition en una aplicación que implementa MVC (pasivo)?
 - NumberAdditionMVC
 - Primero, preguntarse qué o quién es el modelo
 - Qué información almacena el modelo
 - Cada vista crea su propio controlador. A ese controlador se le llama controlador de vista

Ejemplo - NumberAddition

En el proyecto final, habrá tres paquetes:

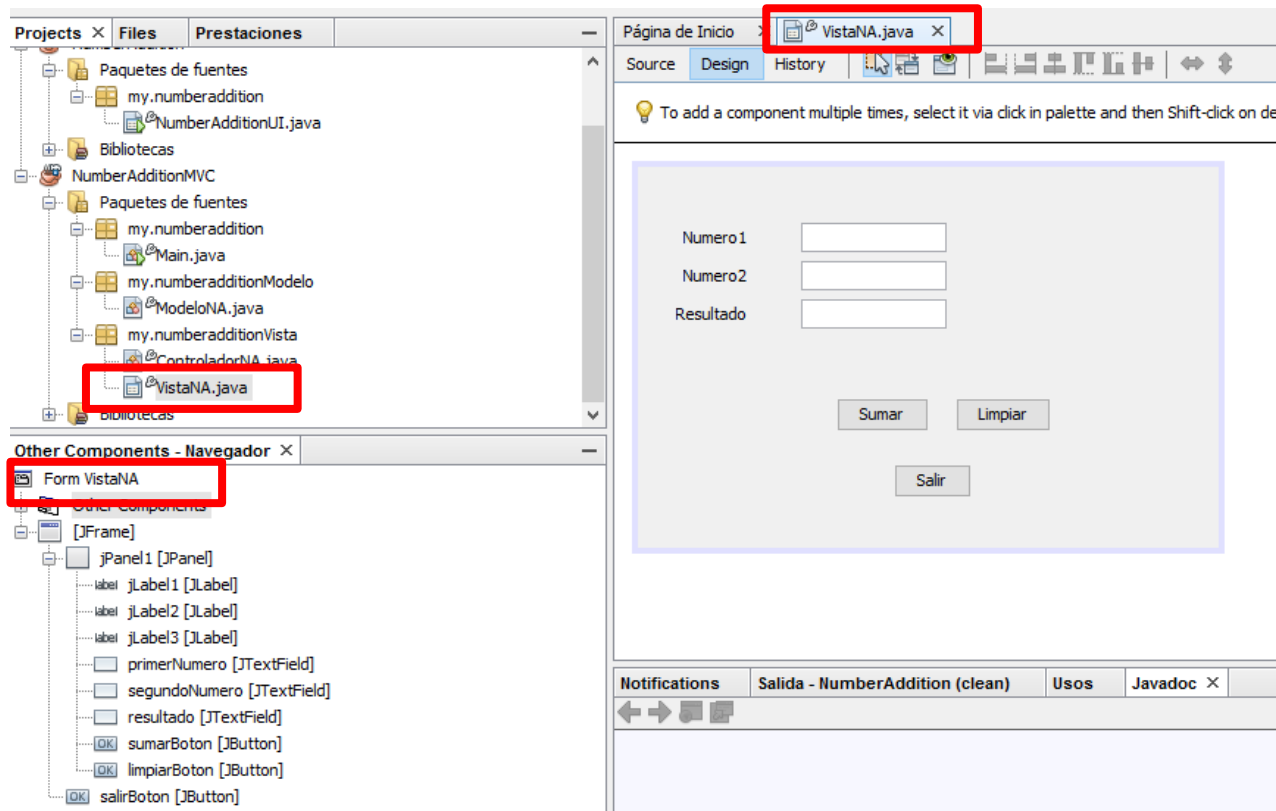
1. Uno contendrá la clase main.
2. El otro, las clases del modelo.
3. Uno más, la clase de la vista y de su controlador.



Ejemplo - NumberAddition

1.- La vista será la clase creada por NetBeans

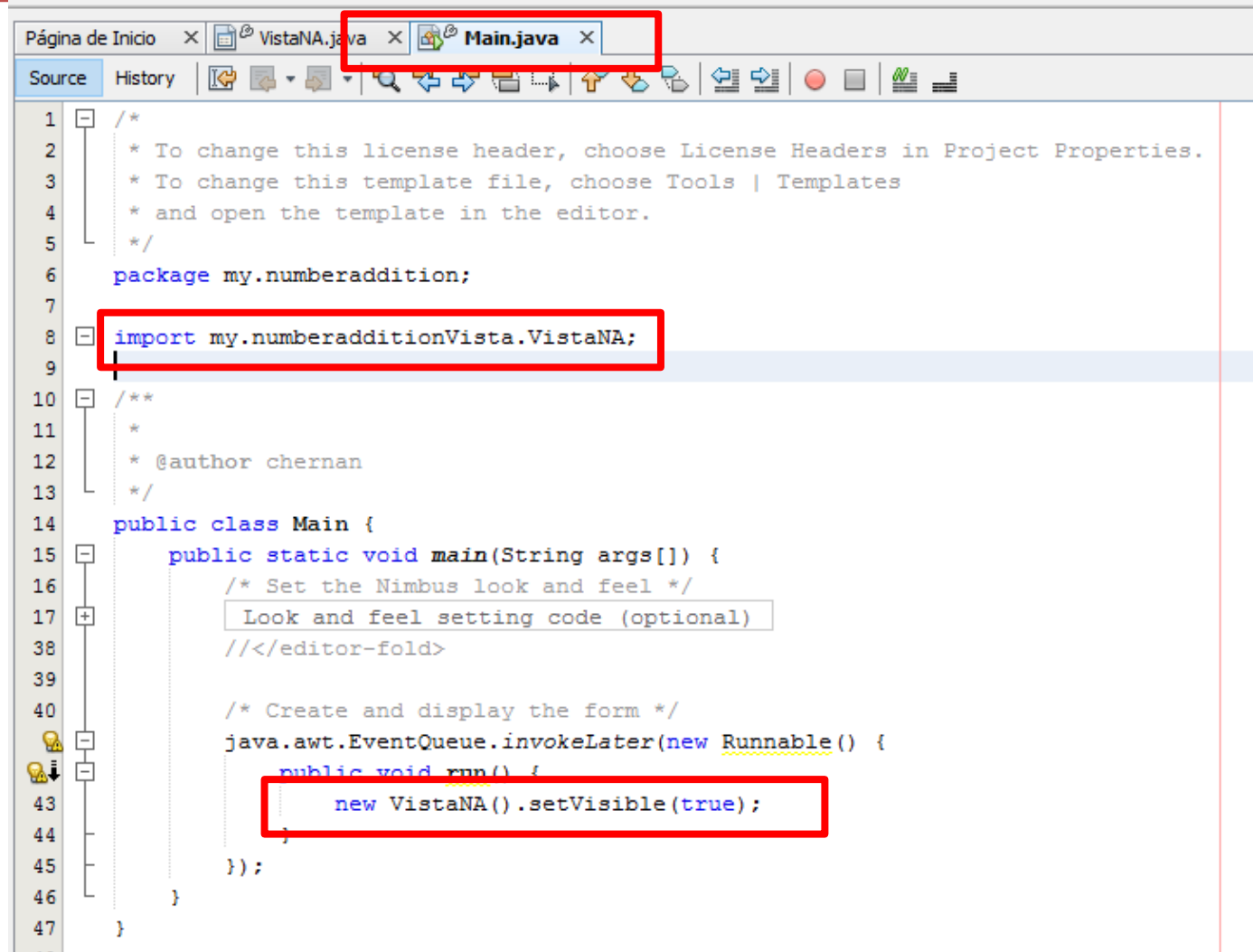
La vista contiene el código para generar la interfaz. No es necesario tocar este código. Se puede renombrar la clase: **VistaNA.java**



Ejemplo - NumberAddition

2.- Se crea una clase Main(), que contenga la función main().

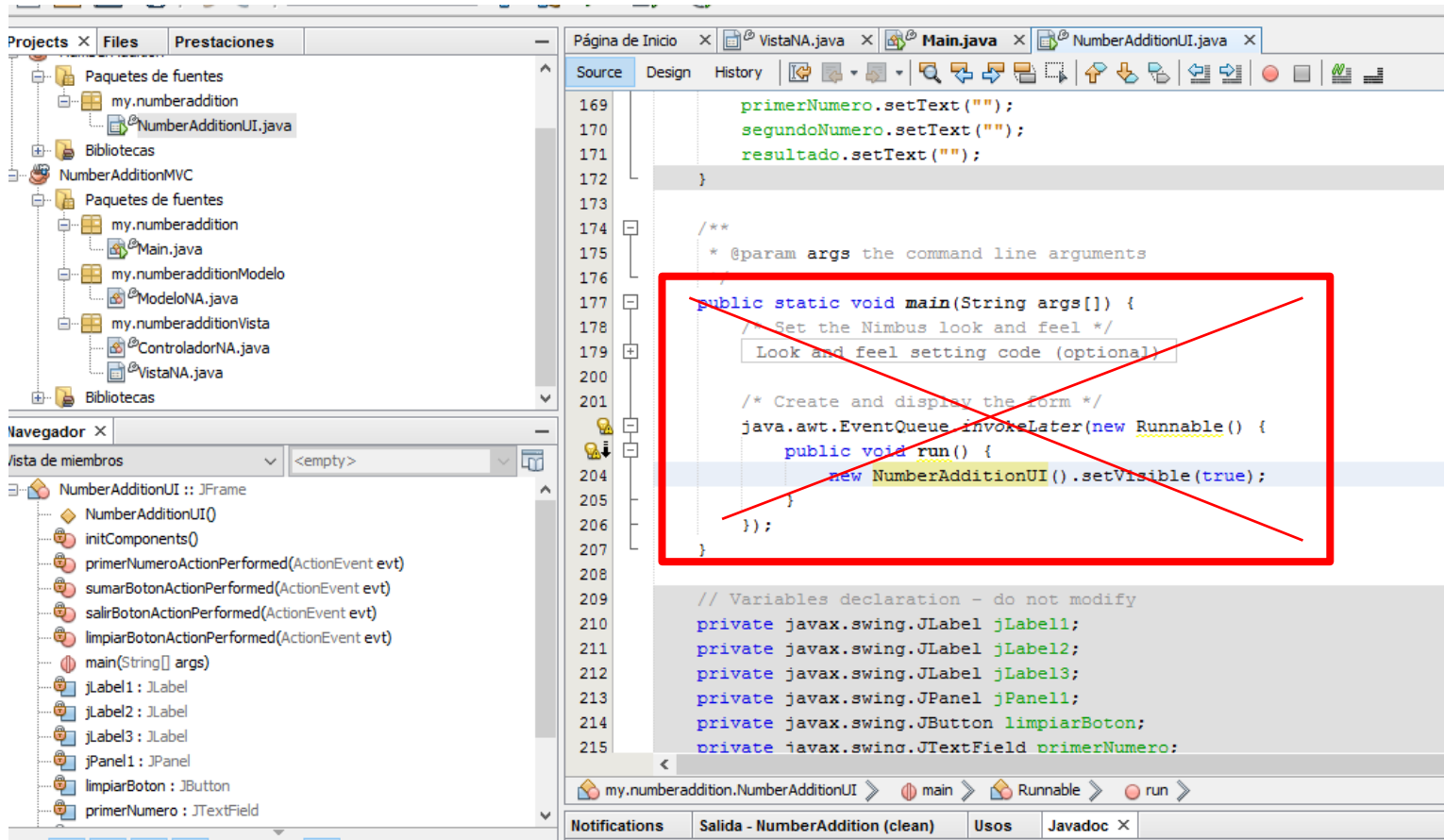
... y se elimina de la vista...



```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package my.numberaddition;
7
8  import my.numberadditionVista.VistaNA;
9
10 /**
11  *
12  * @author chernan
13  */
14 public class Main {
15     public static void main(String args[]) {
16         /* Set the Nimbus look and feel */
17         Look and feel setting code (optional)
18         //</editor-fold>
19
20         /* Create and display the form */
21         java.awt.EventQueue.invokeLater(new Runnable() {
22             public void run() {
23                 new VistaNA().setVisible(true);
24             }
25         });
26     }
27 }
```

Ejemplo - NumberAddition

... y se elimina de la vista



Ejemplo - NumberAddition

3.- Se crea una clase “Controlador”

e.g.: ControladorNA

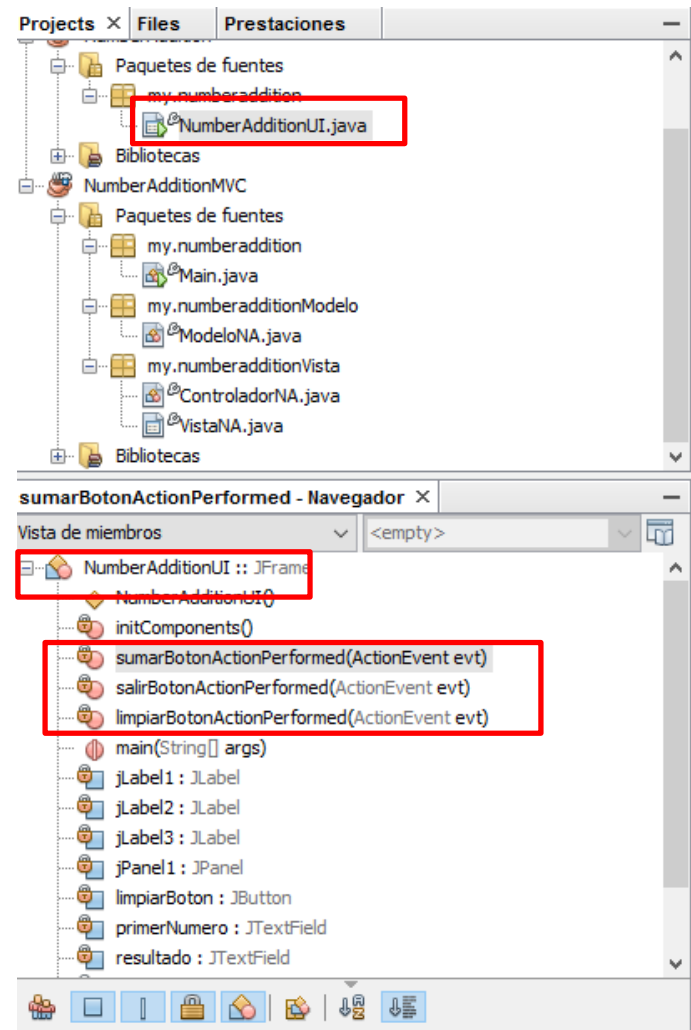
Contiene una referencia a la vista y al modelo.

El constructor recibe vista y modelo como parámetro, para establecer el enlace entre la vista y el controlador

```
6 package my.numberadditionVista;
7
8 import my.numberadditionModelo.ModeloNA;
9
10 /**
11  *
12  * @author chernan
13  */
14 public class ControladorNA {
15     private VistaNA miVista;
16     private ModeloNA miModelo;
17
18     public ControladorNA(VistaNA v, ModeloNA m) {
19         miVista = v;
20         miModelo = m;
21     }
22
23     public void ...
```

Ejemplo - NumberAddition

Además, en el controlador hay que definir métodos para manejar los eventos que estaban ya definidos en la única clase existente (métodos lanzados por los manejadores definidos en Java).

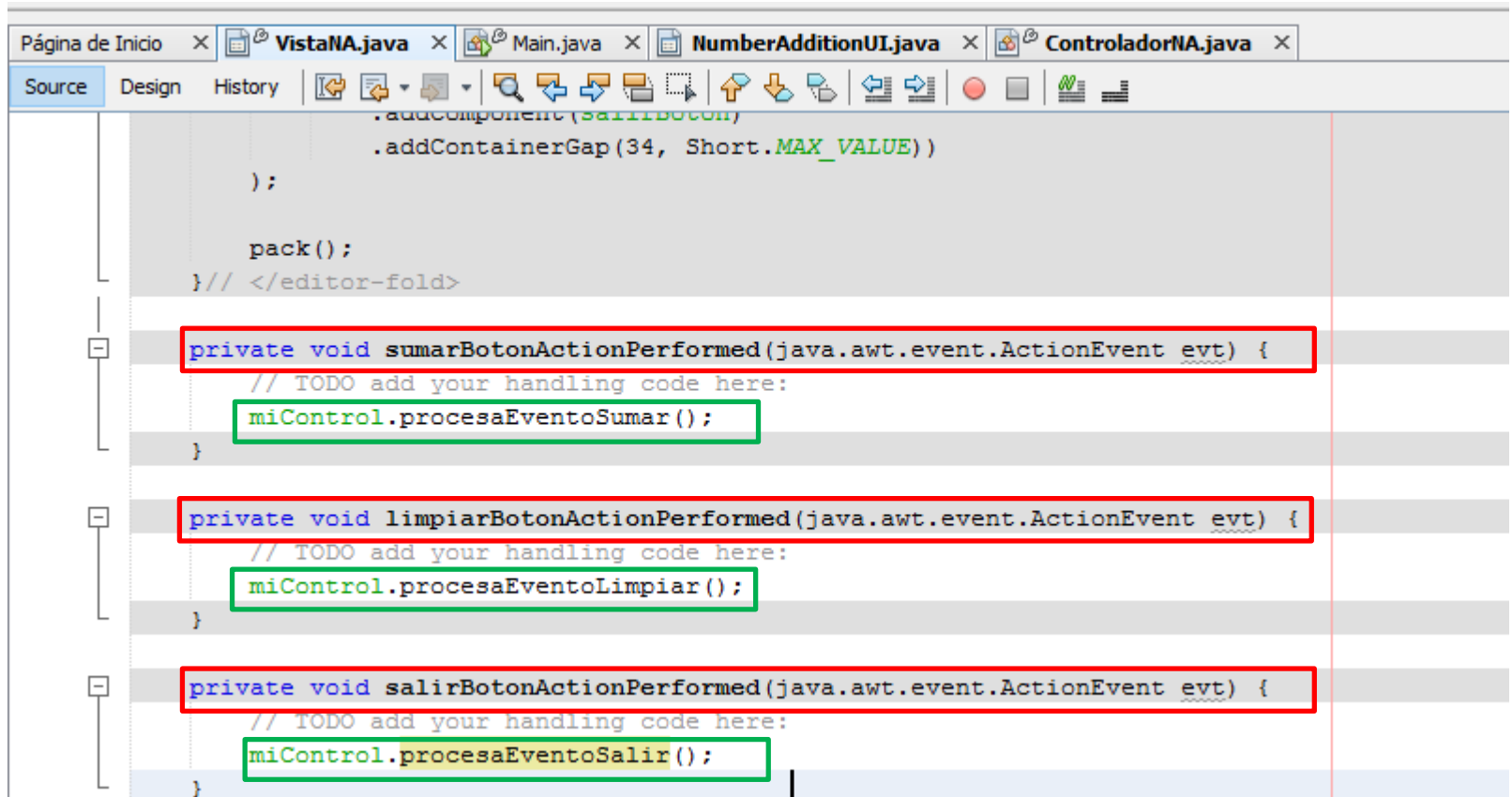


Ejemplo - NumberAddition

```
143
144 private void sumarBotonActionPerformed(java.awt.event.ActionEvent evt) {
145     float num1, num2, result;
146
147     num1 = Float.parseFloat(primerNumero.getText());
148     num2 = Float.parseFloat(segundoNumero.getText());
149
150     result = num1 + num2;
151     resultado.setText(String.valueOf(result));
152 }
153
154 private void salirBotonActionPerformed(java.awt.event.ActionEvent evt) {
155     System.exit(0);
156 }
157
158 private void limpiarBotonActionPerformed(java.awt.event.ActionEvent evt) {
159     primerNumero.setText("");
160     segundoNumero.setText("");
161     resultado.setText("");
162 }
163
164 /**
165  * @param args the command line arguments
```

Ejemplo - NumberAddition

En la vista quedan:



```
Página de Inicio x VistaNA.java x Main.java x NumberAdditionUI.java x ControladorNA.java x
Source Design History
.addComponent(salirBoton)
.addContainerGap(34, Short.MAX_VALUE))
);
pack();
} // </editor-fold>

private void sumarBotonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    miControl.procesaEventoSumar();
}

private void limpiarBotonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    miControl.procesaEventoLimpiar();
}

private void salirBotonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    miControl.procesaEventoSalir();
}
```


Ejemplo - NumberAddition

En el controlador aparecen como:

The screenshot displays an IDE interface with the following components:

- Projects Panel:** Shows the project structure for 'my.numberaddition', including 'NumberAdditionUI.java' and 'Bibliotecas'.
- Navegador Panel:** Shows the 'ControladorNA' class with its members: 'miVista : VistaNA', 'miModelo : ModeloNA', and three methods: 'procesaEventoSumar()', 'procesaEventoLimpiar()', and 'procesaEventoSalir()'.
- Source Editor:** Displays the code for 'ControladorNA.java'. The methods 'procesaEventoSumar()', 'procesaEventoLimpiar()', and 'procesaEventoSalir()' are highlighted with green boxes. The code for 'procesaEventoSumar()' is as follows:

```
19     miVista = v;  
20     miModelo = m;  
21 }  
22  
23 public void procesaEventoSumar() {  
24     //obtener num1  
25     float numero1 = Float.parseFloat(miVista.getNumero1());  
26     //num2  
27     float numero2 = Float.parseFloat(miVista.getNumero2());  
28     //sumar  
29     float res = numero1+numero2;  
30     //el modelo se modifica  
31     miModelo.setResult(res);  
32     //la vista se actualiza  
33     miVista.setResultado();  
34 }  
35  
36 public void procesaEventoLimpiar() {  
37 }  
38  
39  
40 public void procesaEventoSalir() {  
41 }  
42 }  
43 }  
44 }
```
- Notifications Panel:** Shows 'Salida - NumberAddition (clean)' and 'Javadoc'.

Ejemplo - NumberAddition

- Hemos conseguido separar responsabilidades.
- Hemos conseguido independencia de la implementación de la interfaz.
- El código del controlador no depende de los componentes de la vista (JTextField, etc.).
- El controlador es independiente del tipo de interfaz de usuario

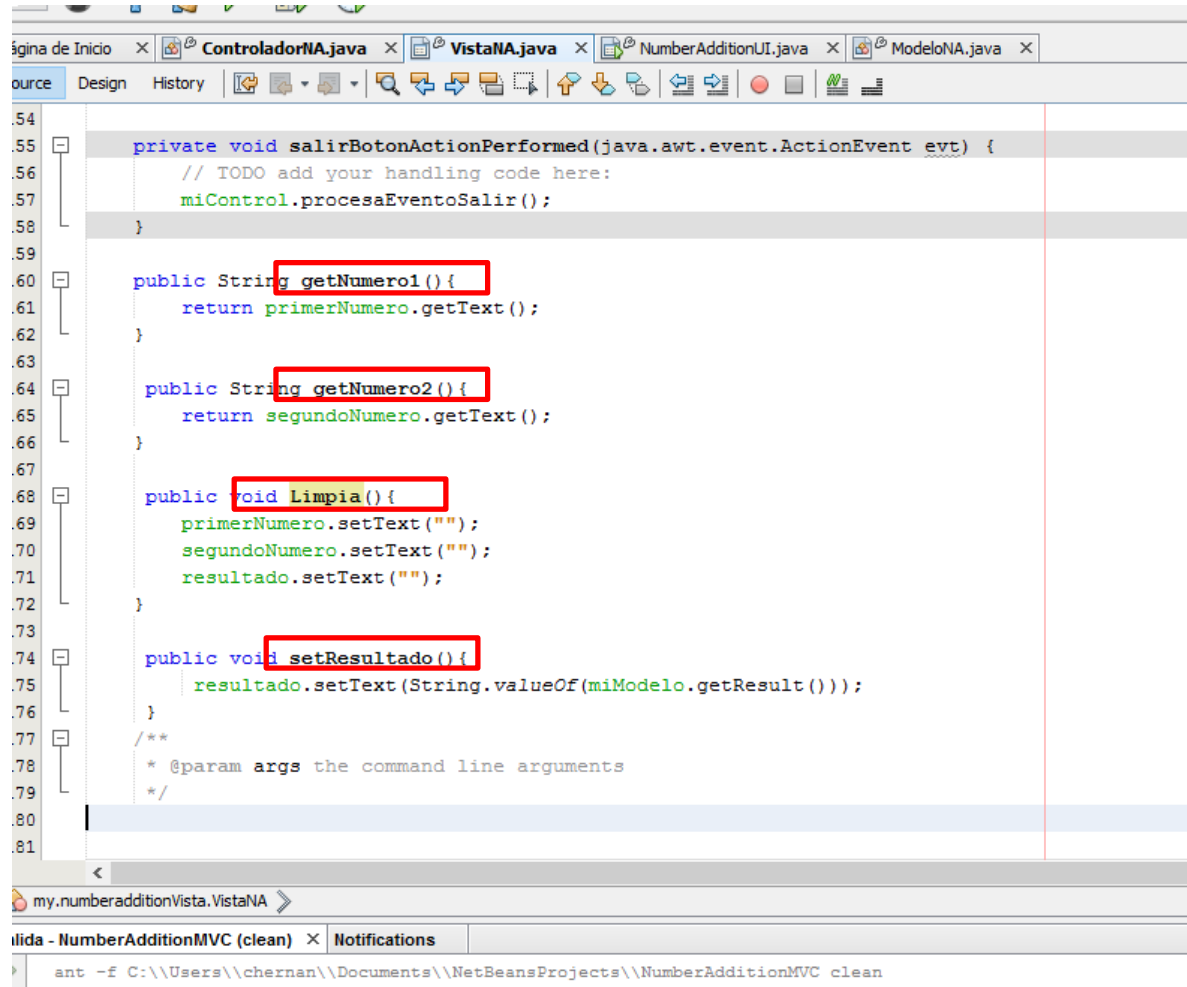
Ejemplo – NumberAddition

Hemos creado nuevos métodos en la vista que realizan las operaciones sin nombrar los componentes que las ofrecen.

□ E.g.:

- `String getNumero1 (); String getNumero2();`
- `void setResultado();`
- `Void Limpia();`
- `void VaciaNum1(); void VaciaNum2(); void VaciaNum3();`

Ejemplo – NumberAddition



```
.54
.55 private void salirBotonActionPerformed(java.awt.event.ActionEvent evt) {
.56     // TODO add your handling code here:
.57     miControl.procesaEventoSalir();
.58 }
.59
.60 public String getNumero1() {
.61     return primerNumero.getText();
.62 }
.63
.64 public String getNumero2() {
.65     return segundoNumero.getText();
.66 }
.67
.68 public void Limpia() {
.69     primerNumero.setText("");
.70     segundoNumero.setText("");
.71     resultado.setText("");
.72 }
.73
.74 public void setResultado() {
.75     resultado.setText(String.valueOf(miModelo.getResult()));
.76 }
.77 /**
.78  * @param args the command line arguments
.79  */
.80
.81
```

my.numberadditionVista.VistaNA

NumberAdditionMVC (clean) × Notifications

ant -f C:\Users\chernan\Documents\NetBeansProjects\NumberAdditionMVC clean

Ejemplo - NumberAddition

Tarea:

- Realizar la conversión de NumberAddition a Number_Addition_MVC