

Desenvolupament Interfície

Interacció amb l'usuari: Events i Listeners



Contents

1	Introducció	2
2	Interactuar amb Events	3
3	ActionEvent	8
4	Components	10
5	Model Vista-Controlador	15
5.1	Programació per capes	16



1 Introducció

Fins ara, he realitzat aplicacions que simplement les llançàvem i aquestes realitzaven la tasca que havíem programat, sense cap possibilitat d'interacció amb ella, de modificar el seu comportament una vegada llançada.

Per poder modificar aquest comportament i tindre capacitat de modifica el comportament de l'aplicació mitjançant la interacció amb l'usuari, cal afegir algun element nou. En aquest cas es el tractament dels Events que es produeixen per la interacció de l'usuari al polsar un botó, polsar alguna tecla, moure el ratolí,... i en funció de l'Event produït realitzar l'acció pertinent.

Al interactuar amb l'aplicació, l'usuari:

- Acciona components (ActionEvent).
- L'usuari polsa un botó.
- L'usuari acaba d'introduir text en un camp i presiona ENTER.
- L'usuari selecciona un element d'una llista polsant el desitjat (o menú)
- Polsar o soltar botons del ratolí (MouseEvent).
- Minimitzar, tancar o manipular la finestra (WindowEvent).
- Escriure amb el teclat (KeyEvent).

Quan l'usuari d'un programa o applet mou el ratolí, presiona un polsador o polsa una tecla, genera un event (actionEvent).

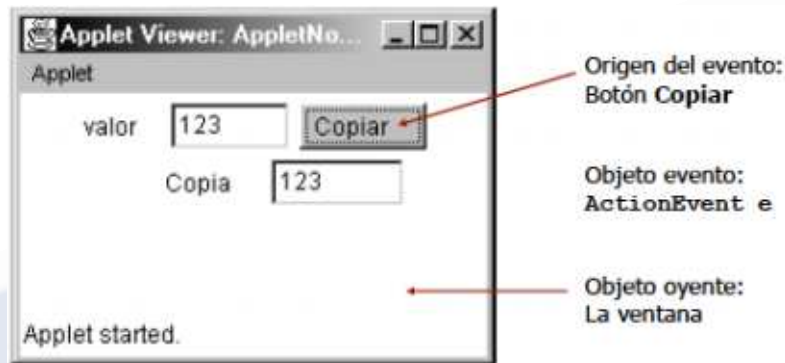
Els events son objectes de determinades classes, normalment un objecte d'alguna subclasse de EventObject que indica:

- L'element que va accionar l'usuari.
- La identificació de l'event que indica el tipus d'event.
- La posició del ratolí en el moment de la interacció.
- Teclades addicionals polsades per l'usuari, com la tecla Control, la tecla Shift,...

Acció	Objecte d'origen	Tipus d'event
Polsar un botó	Jbutton	ActionEvent
Canvi de Text	JTextComponent	TextEvent
Polsar Enter en un quadre de text	JTextField	ActionEvent
Selecció d'un element	JComboBox, JList	ItemEvent, ActionEvent, ListSelectionEvent
Polsar una casella de verificació	JCheckBox	ActionEvent, ItemEvent
Polsar un botó de radio	JRadioButton	ActionEvent, ItemEvent
Seleccionar una opció de menu	JMenuItem	ActionEvent
Moure la barra de desplaçament	JScrollBar	AdjustementEvent
Obrir, tancar, minimitzar la finestra	JWindow	WindowEvent

2 Interactuar amb Events

En l'exemple de la figura, volem que es còpie el que hi ha al JTextField valor al JTextField còpia quan es polsa el botó «copiar». Per a realitzar l'operació, haurem de capturar l'event que es produeix al pulsar el botó «Copiar» i en eixe event copiar el contingut del quadre de text «valor» i copiar-lo a «Copia».



Crearem un manipulador d'events en dos passos:

1. Definirem una classe específica que realitzi la funció d'escollar els events i que implemente el mètode actionPerformed();

```
class MiOyente implements ActionListener{
    public void actionPerformed(){
        //Aci es respon a l'event.
    }
}
```

2. Registrem un exemplar d'oient de components:

```
component.addActionListener(exemplar de miOyente);
```

```
\begin{BVerbatim}
class MiGui extends JFrame {
    ...
    public void MiGui() {
        // Registrem els components interesats
        // en respondre als events ...
        componente.addActionListener(new MiOyente());
        //S'afegeix l'oient especificat (exemplar de MiOyente)
        //a la llista de oients per a rebre events d'acció
        //(ActionEvent) des d'eixe component.
    }
    ...
    class MiOyente implements ActionListener {
        public actionPerformed(ActionEvent e) {
            ...
        }
    }
}
```

```

        // Ací es respon a l'event
        //a l'exemple inserirem el codi per copiar el
        //valor d'un quadre
        //de text a l'altre.
    }
}

```

En aquest cas, vinculem el nou `MiOyente` a la nostra finestra que capturarà aquells events que es puguin produir. Després segons el tipus d'event que siga produït es cridarà un dels mètodes que redefinirem, en aquest cas el `actionPerformed()` rebent com a paràmetre el `ActionEvent` e amb tots els valors que hem comentat abans. Dins d'aquest mètode, ficarem el codi que desitgem que s'execute al produir-se l'event.



Altra forma

Altres dos maneres de definir els events serien aquestes:

```

        boto1.addActionListener(new ActionListener(){
            public void actionPerformed (ActionEvent e){
                //Accions
            }
        });

        ActionListener al=new ActionListener(){
            public void actionPerformed (ActionEvent e){
                //Accions
            }
        };
        boto1.addActionListener(al);

```

1. BotoAmbPitit

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Gui09 extends JFrame {
    JButton boton;

    public Gui09() {
        boton = new JButton("Pulsa!");
        add(boton);
        boton.addActionListener(new OyenteBoton());
        setSize(100, 100);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    ...
}

class OyenteBoton implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        Toolkit.getDefaultToolkit().beep();
    }
}

```



Una clase
de oyente
específica

Gui09.java

- (a) Afegim el botó a la nostra finestra i li vinculem el actionListener per que es dispare al produir-se en aquest cas la pulsació del botó.
- (b) Després al manejador de l'event, dins l'event actionPerformed() produïm un petit mitjançant l'altaveu del sistema.

Si en el nostre cas no s'escolta, podem afegir-li una instrucció per que mostre per consola un missatge i verificar que si funciona: `System.out.println("Boto polsat");` Tin en conter que al exemple falta crear la classe main per a executar el programa.

Exercici 1

- (a) Fes l'exemple anterior amb les altres dos formes de definir els events.

2. Copiar un valor

El següent exemple es preten copiar el text d'un camp de text a altre quan el botó es presionat.

Es pot observar com després d'afegir els components li vinculem al botó el Listener per a que en el moment que es produeix l'event al botó de copiar realitzi les accions que tenim definides al actionPerformed() del ActionListener. Arrepleguem a la variable valor el que hi ha introduït al quadre de text i ho escrivim al camp de resultat.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```



```
public class Gui10 extends JFrame {

    JButton botonCopiar;
    JTextField campoValor, resultado;

    public Gui10() {
        setLayout(new FlowLayout());
        add(new JLabel("Valor "));
        campoValor = new JTextField(5);
        add(campoValor);
        botonCopiar = new JButton("Copiar");
        add(botonCopiar);
        botonCopiar.addActionListener(new OyenteBoton());
        add(new JLabel("Copia "));
        resultado = new JTextField(6);
        setSize(400, 100);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui10 ventana = new Gui10();
    }

    class OyenteBoton implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String valor = campoValor.getText();
            resultado.setText(valor);
        }
    }
}
```


3. Contador de pulsacions

En aquest exemple veiem que cada vegada que polsem el botó, anem incrementant el comptador que hi ha al costat. Per fer aquest efecte, al actionPerformed, anem canviant el valor que mostra un Label, per l'actualitzat.


```

public class Gui11 extends JFrame {
    ...
    public Gui11() {
        boton1 = new JButton("PULSA");
        labell1 = new JLabel("Pulsaciones: 0");
        add(boton1);
        add(labell1);
        setLayout(new FlowLayout());
        boton1.addActionListener(new OyenteBotonPulsaciones());
        ...
    }
    ...
    class OyenteBotonPulsaciones implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            contador++;
            labell1.setText("Pulsaciones: " + contador);
        }
    }
}

```



Gui11.java

4. Canvi de color

Al polsar un dels dos botons, desitgem que canvie el color de fons de la finestra i el que fem es vincular a cada botó una instancia d'un oient diferent. Que el que fa es canviar el color de fons a blau o roig.


```

import javax.swing.*;
import java.awt.*; import java.awt.event.*;

public class Gui12 extends JFrame {
    JButton rojo = new JButton("Rojo");
    JButton azul = new JButton("Azul");
    Container p;
    public Gui12() {
        super("Color de fondo");
        p = this.getContentPane();
        setLayout(new FlowLayout());
        add(rojo);
        add(azul);
        rojo.addActionListener(new OyenteRojo());
        azul.addActionListener(new OyenteAzul());
        setSize(200, 200);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String args[]) {
        Gui12 ventana = new Gui12();
    }
    class OyenteRojo implements ActionListener {
        public void actionPerformed(ActionEvent evento) {
            p.setBackground(Color.red);
        }
    }
    class OyenteAzul implements ActionListener {
        public void actionPerformed(ActionEvent evento) {
            p.setBackground(Color.blue);
        }
    }
}

```



Gui12.java

3 ActionEvent

Per tal de poder diferenciar quin és component que ha produït l'event i no tindre que crear múltiples Listener, podem utilitzar dos mètodes:

1. L'objecte `ActionEvent` és el que rebem com a paràmetre al produir-se un even. Disposa d'un mètode `getActionCommand()` que retorna un objecte amb la informació sobre l'origen del event (en l'exemples que hem vist, el component que causaba el event era el botó).

Per a poder identificar botons individuals, el valor que retorna en el cas dels botons és el de l'etiqueta del botó que ha causat l'event:

```
public void actionPerformed(ActionEvent e) {  
    String s = (String)e.getActionCommand();  
    if(s.equals("Aceptar")) {  
        // Tractament del botó Aceptar  
    }  
    ...  
}
```

En l'exemple anterior s'observa que el redefinir el mètode `actionPerformed` (on van les instruccions de les accions de l'event), s'utilitza el mètode `getActionCommand()` convertint el que retorna a `String` per tal de poder comparar amb el valor de l'etiqueta i poder discriminar possibles botons que coincideixen.

2. El mètode `getSource()` indica l'objecte en el que s'ha produït l'event:

```
public void actionPerformed(ActionEvent e) {  
    if(e.getSource() == lista) {  
        campo1.setText("En la llista.");  
    }  
    else if(e.getSource() == texto) {  
        campo2.setText("En el camp de text.");  
    }  
    else if(e.getSource() == boton) {  
        campo3.setText("Al botó.");  
    }  
}
```

En aquest cas, el valor que retorna el mètode és el nom que li hem posat al component al crearlo.

5. Conversor Euros a pessetes

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class Gui13 extends JFrame {
    private JTextField cantidad;
    private boton1, boton2;

    public Gui13() {
        super("Conversor Euros-Pesetas");
        boton1 = new JButton("A euros");
        boton2 = new JButton("A pesetas");
        cantidad = new JTextField(10);
        JLabel eti2 = new JLabel(new ImageIcon("logo.gif"));
        add(eti2); add(cantidad);
        add(boton1); add(boton2);
        setLayout(new FlowLayout());
        boton1.addActionListener(new OyenteBoton());
        boton2.addActionListener(new OyenteBoton());
        setSize(300, 250);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```



```
public static void main(String args[]) {
    Gui13 ventana = new Gui13();
}

class OyenteBoton implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        Float f = new Float(cantidad.getText());
        float valor = f.floatValue();
        String s = (String)ae.getActionCommand();
        if(s.equals("A euros")) {
            valor = (float) (valor / 166.321);
        }
        else if(s.equals("A pesetas")) {
            valor = (float) (valor * 166.321);
        }
        cantidad.setText(Float.toString(valor));
    }
}
```

La clase OyenteBoton es interna y tiene acceso a los elementos de la clase Contenedora

Gui13

Utilitzant el que hem vist abans, implementem un únic ActionListener i al tindre dos botons el que fem es quan salta l'event, verificar quin dels dos es el que l'ha produït i en funció d'això convertim el valor a euros o a pessetes.

Exercici 2

- (a) No obstant es pot millorar aquest exercici ja que en cap moment sabem si la quantitat que tenim en pantalla son euros o pessetes. Per tant, es proposa que en el cas de realitzar la conversió a pessetes aparega al costat del valor, el símbol "ptes." i a més deshabilitar el botó de conversió a pessetes. Si ho fem amb el de euros, ha d'aparèixer el símbol € al costat del valor i deshabilitar el boto de "a euros". Decidirem si inicialment el valor es d'euros o de pessetes i es dirà Gui13b.java

Activar i desactivar botons

Per a activar i desactivar botons s'utiliza el mètode `setEnabled()` de la següent forma:

- Per a activar el botó, suposant que el botó es diga boton1:
`boton1.setEnabled(true);`
- Per a desactivar el botó, suposant que el botó es diga boton1:
`boton1.setEnabled(false);`

Per facilitar la tasca del programador s'han creat una serie de interfícies que han d'implementar-se quan es requerisca processar alguns d'aquests events. Algunes son:

ActionListener Escolta events de tipus ActionEvent	<code>actionPerformed(ActionEvent)</code>
KeyListener Escolta events de teclat	<code>keyPressed(KeyEvent)</code> <code>keyReleased(KeyEvent)</code> <code>keyTyped(KeyEvent)</code>
MouseListener Escolta events de acció del ratolí (botones)	<code>mouseClicked(MouseEvent)</code> <code>mouseEntered(MouseEvent)</code> <code>mouseExited(MouseEvent)</code> <code>mousePressed(MouseEvent)</code> <code>mouseReleased(MouseEvent)</code>
MouseMotionListener Escolta events de moviment del ratolí	<code>mouseDragged(MouseEvent)</code> <code>mouseMoved(MouseEvent)</code>

4 Components

1. **Botons:** els diferents tipus de botons que tenim són:

- JButton: Botó aïllat. Pot polsar-se però el seu estat no canvia.
- JToggleButton: Botó seleccionable. Quan es polsa el botó canvia el seu estat a seleccionat fins que es polsa de nou
 - isSelected() permet verificar el seu estat.
- JCheckBox: Especialització de JToggleButton que implementa una casilla de verificació. Botó amb estat intern, que canvia l'aparença de forma adequada segons si està o no seleccionat.
- JRadioButton: Especialització de JToggleButton que te sentit dins d'un mateix grup de botons (ButtonGroup) que controla que tan sols un d'ells pot estar marcat.

(a) **Exemple: JButton**

- Constructors:
 - JButton(String text);
 - JButton(String text, Icon icon);
 - JButton(Icon icon);
- Resposta a botons:
 - Implementar la interfaz ActionListener
 - Implementar el mètode actionPerformed(ActionEvent e)

```
boton1 = new JButton("A Euros ");
```

```
boton1.setIcon(new ImageIcon("flag1.gif"));
```



```
boton2 = new JButton(new ImageIcon("flag2.gif"));
```



```
boton3 = new JButton("Botón",new ImageIcon("flag8.gif"));
```



2. **Etiquetes: JLabel** : Servix per a insertar text en la interfície, una imatge o les dos coses.

```
JLabel(String text, int horizontalAlignment)
```

```
JLabel(String text)
```

```
JLabel(Icon icon)
```

```
JLabel(Icon icon, int horizontalAlignment)
```

```
eti1 = new JLabel("Etiqueta de text...");
```

```
eti2 = new JLabel(new ImageIcon("flag8.gif"));
```



3. *TextField*: Camps de text per introduir caràcters

```

TextField(int columns)
TextField(String text)
TextField(String text, int columns)
TextField text1 = new TextField("hola", 10);
Posar text: text1.setText("Adios");

Obtenir text: String str = text1.getText();

Agregar al Panel: p1.add(text1);

```

4. *ComboBox*: Llistes d'elements per a seleccionar un sol valor:

- Creació: `JComboBox ch1 = new JComboBox();`
- Afegir opcions: `ch1.addItem(Object element);`
- Registrar Event: `ch1.addItemListener(objecte);`
- Obtenir selecció:
 - `val = ch1.getSelectedIndex();`
 - `ch1.getItem()`
- Implementar la interfaz `ItemListener`
- Implementar el mètode `itemStateChanged(ItemEvent e)`

```
ch1.addItemListener(new Oyenteltem());
```

```

...
class Oyenteltem implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        ...
    }
}

```

5. *JList*: Llistes d'elements per a seleccionar un o més valors:

- – `JList l1 = new JList();`

- `JList l2 = new JList(Object[]elements);`
- `String[]coses = "Opci1", "Opci2", "Opci3";`
- `Jlist l2 = new Jlist(coses);`
- Registrar event: `l2.addListSelectionListener(oyente);`
- Obtenir selecció:
 - `int[] indices = l2.getSelectedIndices();`
- Implementar la interfaz `ListSelectionListener`
- Implementar el mètode `valueChanged(ListSelectionEvent e)`

```
l.addListSelectionListener(new OyenteLista());

class OyenteLista implements ListSelectionListener {

    public void valueChanged(ListSelectionEvent e) {

        int[] indices = l.getSelectedIndices();
        int i;
        for(i = 0; i < indices.length; i++) {
            ...
        }
    }
}
```

6. *JScrollBar*:



- Creació: `bar1 = new Scrollbar(Scrollbar.HORIZONTAL,0,0,0,100);` Els Valors que tenim en el constructor son:
 - Orientació: horizontal o vertical
 - El valor inicial.
 - El valor del scrollbar, es a dir, el tamany de la barra desplazadora. El valor mínim és 0
 - El valor mínim de la barra
 - El valor màxim de la barra
- Registrar event: `bar1.addAdjustmentListener(oyente);`
- Implementar la interfaz `AdjustmentListener`, Implementar el mètode: `adjustmentValueChanged(AdjustmentEvent e)`


```

bar1.addAdjustmentListener(new OyenteBarra());
...
class OyenteBarra implements AdjustmentListener {
    public void adjustmentValueChanged(AdjustmentEvent e) {
        ... }
}

```

Obtenir valor:

```

int val = bar1.getValue(); // val entre 0 y 100

```

6. **Exemple: ScrollBarComboBox** En aquest exemple podem observar com es crea un JComboBox i a més li afegim la llista d'opcions. Li vinculem el listener per a que reaccione quan ocórrega algun event i fem el mateix amb la barra donant-li uns valors entre 0 i 100. Situem els components dins un panell i només ens resta implementar el que desitgem que ocórrega als events.

```

public class Gui15 extends JFrame {
    Container panel;
    JPanel p1, p2;
    JLabel l1, msg;
    JComboBox chl;
    String[] lista = {"Opción 1", "Opción 2", "Opción 3"};
    JScrollBar bar1;

    public Gui15() {
        super("Controles");
        setLayout(new BorderLayout());
        p1 = new JPanel(new GridLayout(1, 3, 10, 10));
        p1.setBackground(Color.lightGray);
        l1 = new JLabel("Elegir:", Label.RIGHT);
        l1.setBackground(Color.yellow);
        p1.add(l1);
        chl = new JComboBox();
        chl.addItemListener(new OyenteCombo());
        p1.add(chl);
        bar1 = new JScrollBar(Scrollbar.HORIZONTAL, 0, 0, 0, 100);
        /* scroll de 0 a 100*/
        bar1.addAdjustmentListener(new OyenteBarra());
        p1.add(bar1);
        p2 = new JPanel(new BorderLayout());
        p2.setBackground(Color.lightGray);
        msg = new JLabel("Msg:", Label.LEFT);
        msg.setForeground(Color.blue);
        p2.add("North", msg);
        add(p1, "North");
        add(p2, "South");
        setSize(300, 100);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```




```

public static void main(String args[]) {
    Gui15 ventana = new Gui15();
}

class OyenteCombo implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        int ind = chl.getSelectedIndex();
        msg.setText((String)chl.getSelectedItem());
    }
}

class OyenteBarra implements AdjustmentListener {
    public void adjustmentValueChanged(AdjustmentEvent e) {
        int valor = bar1.getValue();
        String cad = "Valor : " + valor;
        msg.setText(cad);
    }
}
}

```

Gui15.java

Al canviar la selecció dins al comboBox, canviem el missatge que es mostra i al modificar la posició o desplaçar la barra, anem canviant el valor numèric que es mostra a la part inferior.

7. Exemple: moviments Ratolí

En aquest exemple el que podem observar afegim un listener per accionar quan es produeix alguna acció per part del ratolí i ho mostrem a un JList que va llistat totes les accions del ratolí que es van produint.

Podent-se completar l'exercici afegint un MouseListener per afegir tipus de events. Es recomana una vegada comprovat el funcionament, comentar la línia de "moviendo" ja que a poc que es moga el ratolí inundarà la llista i no deixarà vore la resta d'events.

```

public class Gui17 extends JFrame {
    JButton boton;
    List lista;
    Container panel;

    public Gui17() {
        ...
        this.addMouseMotionListener(new OyenteMover());
        ...
    }
    ...
}

class OyenteMover implements MouseMotionListener {
    public void mouseDragged(MouseEvent e) {
        lista.add("arrastrando..");
    }
    public void mouseMoved(MouseEvent e) {
        lista.add("moviendo..");
    }
}

```



Gui17.java

```

class OyenteRaton implements MouseListener {
    public void mouseClicked(MouseEvent e) {
        lista.add("click..");
    }
    public void mouseEntered(MouseEvent e) {
        lista.add("enter..");
    }
    public void mouseExited(MouseEvent e) {
        lista.add("exit..");
    }
    public void mousePressed(MouseEvent e) {
        lista.add("pulsar..");
    }
    public void mouseReleased(MouseEvent e) {
        lista.add("soltar..");
    }
}

```



5 Model Vista-Controlador

El Model Vista Controlador (MVC) es un patró d'arquitectura de software que separa les dades d'una aplicació, la interfície d'usuari, i la lògica de negoci en tres components:

- **Model:** Esta es la representació específica de la informació amb la qual el sistema opera. També es la que interactua directament amb la base de dades
- **Vista:** Este presenta el modelo en un format adequat per interactuar amb el sistema, usualment la interfície d'usuari.
- **Controlador:** Respon als events, es a dir, a les accions de l'usuari, i invoca peticions al modelo i, probablement, a la vista.

El flux que segueix la implementació del model:

1. L'Usuari interactua amb el sistema mitjançant la Vista d'Usuari (GUI). Per exemple pressionant botons, introduint text, movent el cursor per la pantalla, etc.

2. El controlador és el que reb totes estes accions provocades per la interacció en la interfície i els event es desencadenen.
3. El controlador accedix al Model. Depenent de l'event sol·licitat, es realitzarà una actualització (registre, edició o eliminació d'informació) en la base de dades o sol·licitant informació (una consulta per exemple).
4. La informació és processada en model-controlador i és el controlador l'encarregat de generar una eixida per a l'usuari.
5. Aleshores la Interfície d'usuari (Vista) queda en espera d'una acció de l'usuari del sistema per repetir el cicle de nou.

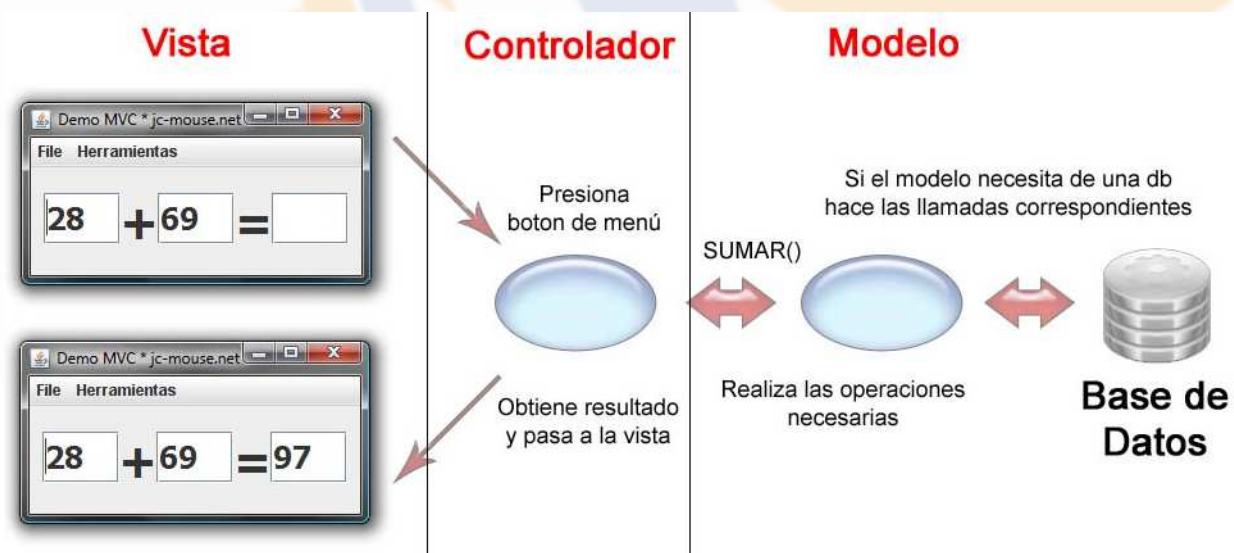
5.1 Programació per capes

El propòsit es fer una desagregació més especialitzada de las capas abans vistes en el modelo MVC que componen un sistema software o una arquitectura client-servidor. La més utilitzada és la de 3 capes, en que bàsicament venen a coincidir a les capes de MVC:

- **Capa de presentació:** Esta es la que veu l'usuari
- **Capa de negoci:** es on resideixen els programes que se executen, se reben les peticions de l'usuari i se envien les respostes després del proces.
- **Capa de dades:** es on resideixen les dades i és la encarregada d'accedir als ells.

Exemple:

En el següent exemple es pot veure la implementació del patró MVC que coincideix amb el modelo de 3 capes:



1. Començant per el model, seria el següent codi:

```

package model;
/**
 * @web http://jc-mouse.blogspot.com/
 * @author Mouse
 * En esta clase se guarda la lógica del sistema, para
 * consiste en una SUMA
 */
public class modelo {
    //Variables
    private int valor1=28;
    private int valor2=69;
    private int total = sumar();

    public modelo(){

    }

    public void set_valor1(int val){
        this.valor1=val;
    }

    public int get_valor1(){
        return this.valor1;
    }

    public void set_valor2(int val){
        this.valor2=val;
    }
}

```

En aquest cas, el model és senzill, ja que l'única cosa que fa es sumar amb els mètodes de llegir i establir valor a allò que s'ha de sumar.

2. Continuarem per la Vista, que serà l'interfície que es pot veure a dibuix anterior.
3. El controlador, seria l'encarregat de respondre als events, si tenim un botó en l'interfície per a que es calcule la suma de dos valors, en esta part del programa, s'hauria de fer el procés de la suma.