



GUI { java swing }

- La interfaz gráfica de usuario (GUI) juega un papel crucial en la experiencia del usuario. Java Swing es una de las bibliotecas más populares para crear GUIs en Java.
- Java Swing es una biblioteca de GUI que forma parte del JFC (Java Foundation Classes). Proporciona un conjunto de componentes ligeros y personalizables para construir interfaces gráficas de usuario.

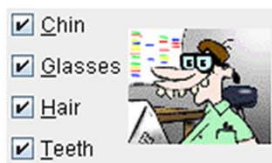
GUI { java swing }

- El diseño de una GUI en Swing se basa en el uso de contenedores y componentes.
- Los contenedores, como JFrame y JPanel, son responsables de organizar y contener otros componentes como botones, etiquetas y campos de texto.
- Los diseños (layouts) determinan cómo se colocan estos componentes dentro de los contenedores.

GUI { java swing }



[JButton](#)



[JCheckBox](#)



[JComboBox](#)



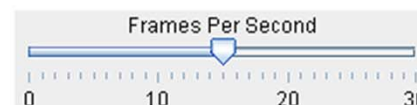
[JList](#)



[JMenu](#)



[JRadioButton](#)



[JSlider](#)



[JSpinner](#)

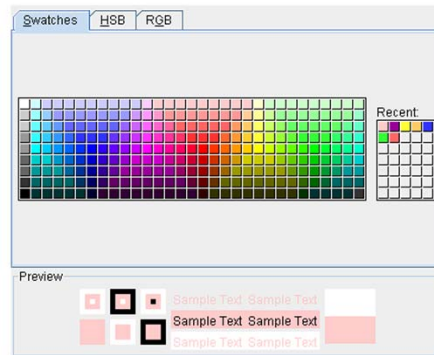


[JTextField](#)



[JPasswordField](#)

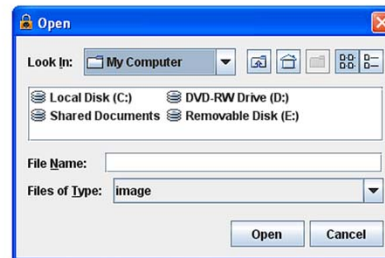
GUI { java swing }



[JColorChooser](#)



[JEditorPane](#) and [JTextPane](#)



[JFileChooser](#)

Host	User	Password	Last Modified
Blocca Games	Freddy	#asf6Awz6	Mar 16, 2006
zabble	ichabod	Tazb1348Z	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasV5411fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail...	bKzjADF78l	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbA124%z	Feb 22, 2006

[JTable](#)

This is an editable *JTextArea*. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.

[JTextArea](#)



[JTree](#)

GUI { java swing }



[JLabel](#)



[JProgressBar](#)



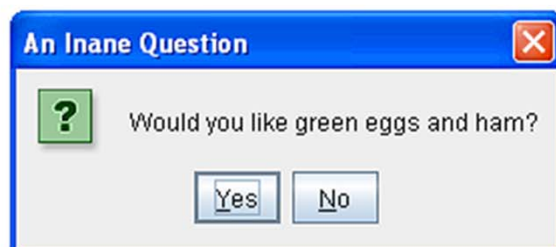
[JSeparator](#)



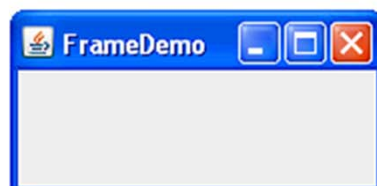
[JToolTip](#)



[JApplet](#)

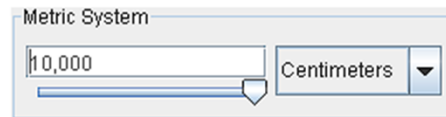


[JDialog](#)

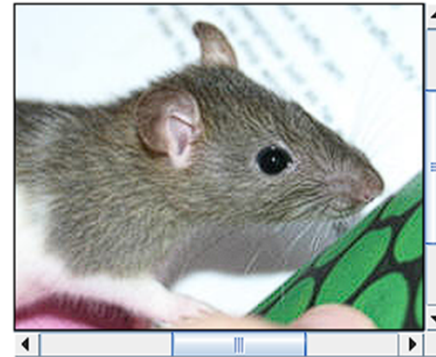


[JFrame](#)

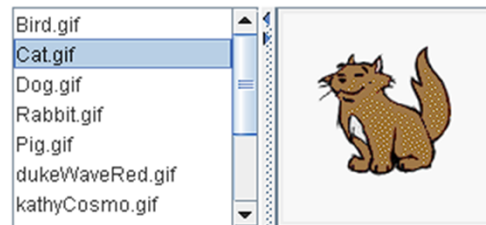
GUI { java swing }



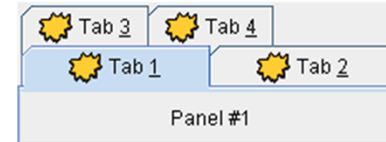
[JPanel](#)



[JScrollPane](#)



[JSplitPane](#)

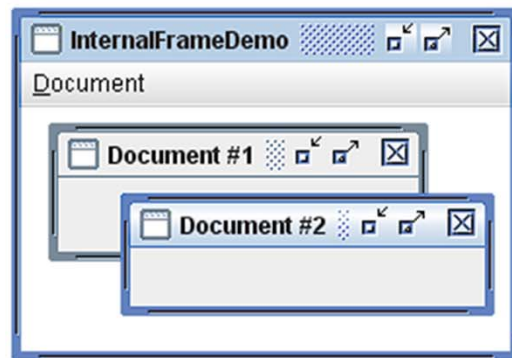


[JTabbedPane](#)

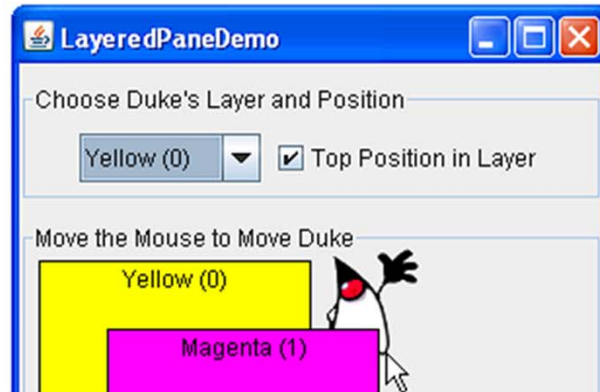


[JToolBar](#)

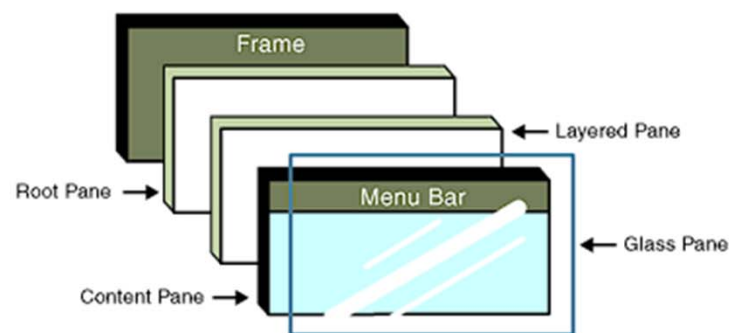
GUI { java swing }



[JInternalFrame](#)



[JLayeredPane](#)



[Root pane](#)

Ejemplo

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.GridLayout;

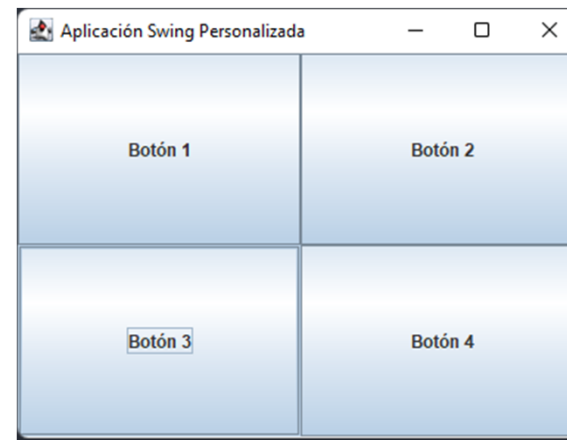
public class CustomSwingApp extends JFrame {
    public CustomSwingApp() {
        setTitle("Aplicación Swing Personalizada");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        initUI();
    }

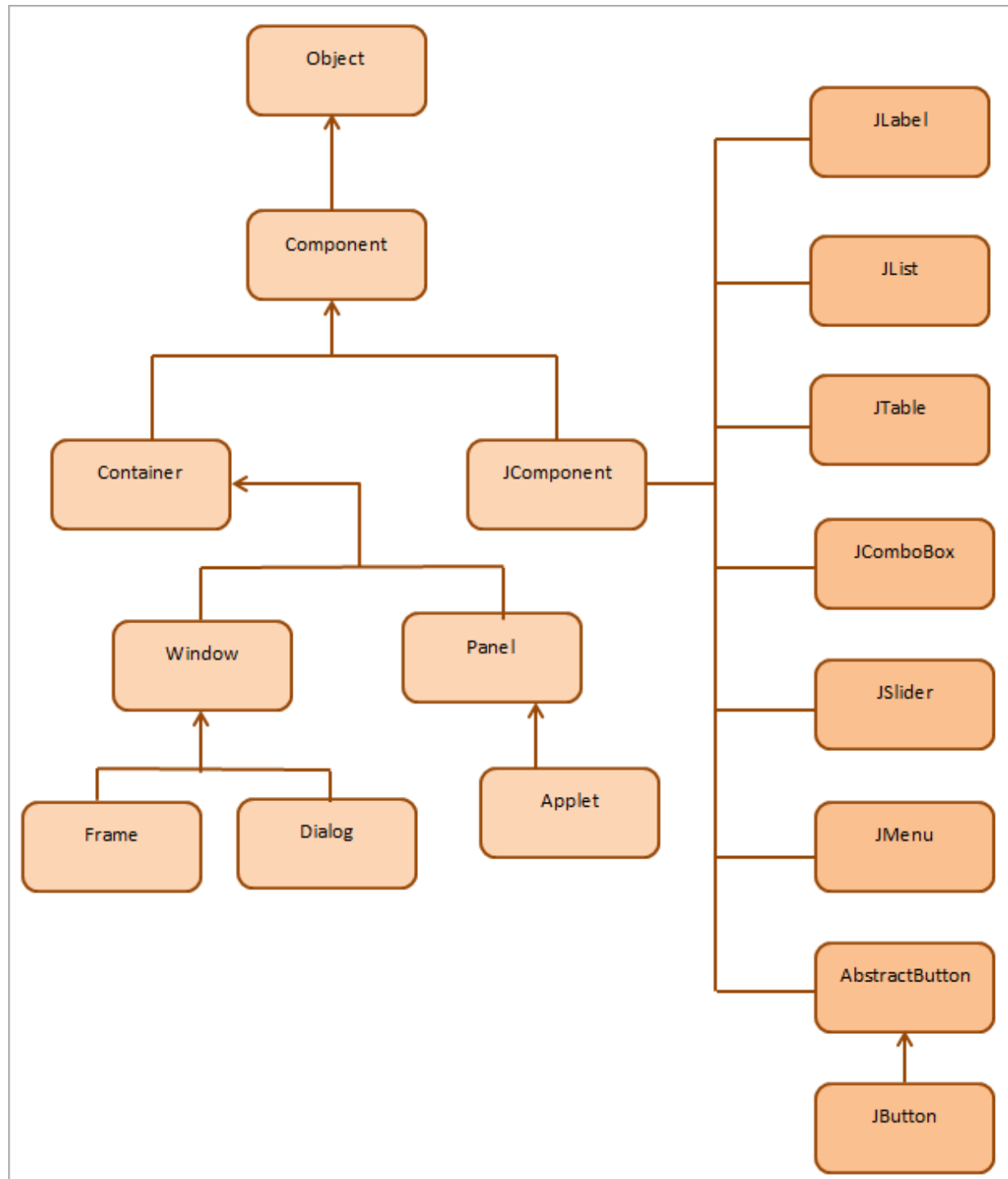
    private void initUI() {
        JPanel panel = new JPanel(new GridLayout(2, 2));
        JButton button1 = new JButton("Botón 1");
        JButton button2 = new JButton("Botón 2");
        JButton button3 = new JButton("Botón 3");
        JButton button4 = new JButton("Botón 4");

        panel.add(button1);
        panel.add(button2);
        panel.add(button3);
        panel.add(button4);


        add(panel);
    }

    public static void main(String[] args) {
        CustomSwingApp app = new CustomSwingApp();
        app.setVisible(true);
    }
}
```






Para el eclipse




WindowBuilder Current

WindowBuilder is composed of SWT Designer and Swing Designer and makes it very easy to create Java GUI applications without spending a lot of time writing code.... [more info](#)

by [Eclipse Foundation](#), EPL
[SWT](#) [swing](#) [wysiwyg](#) [graphical](#) [WindowBuilder](#)


★ 1057  Installs: **1.16M** (7,368 last month) Change ▼



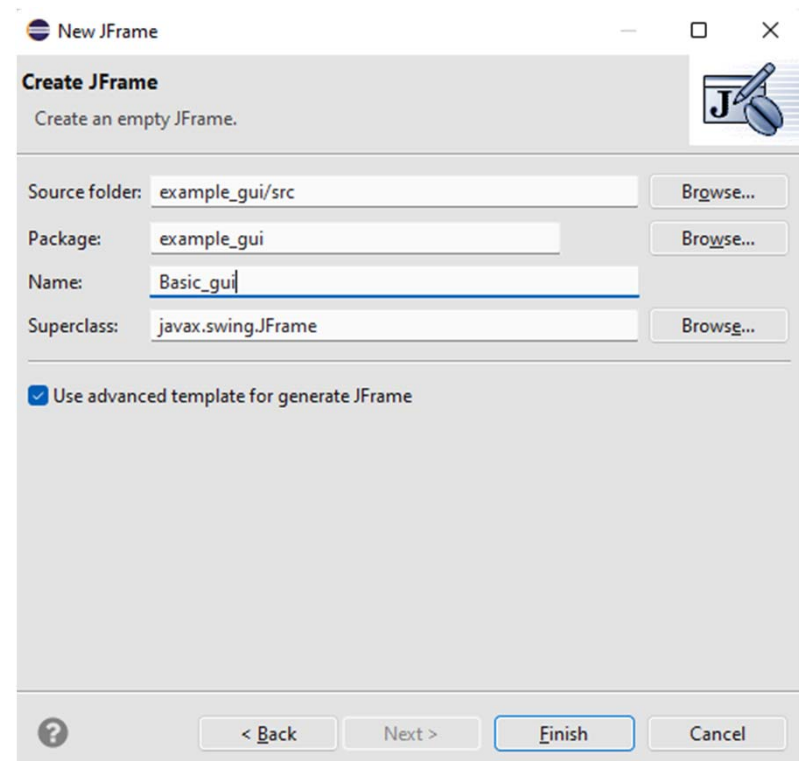
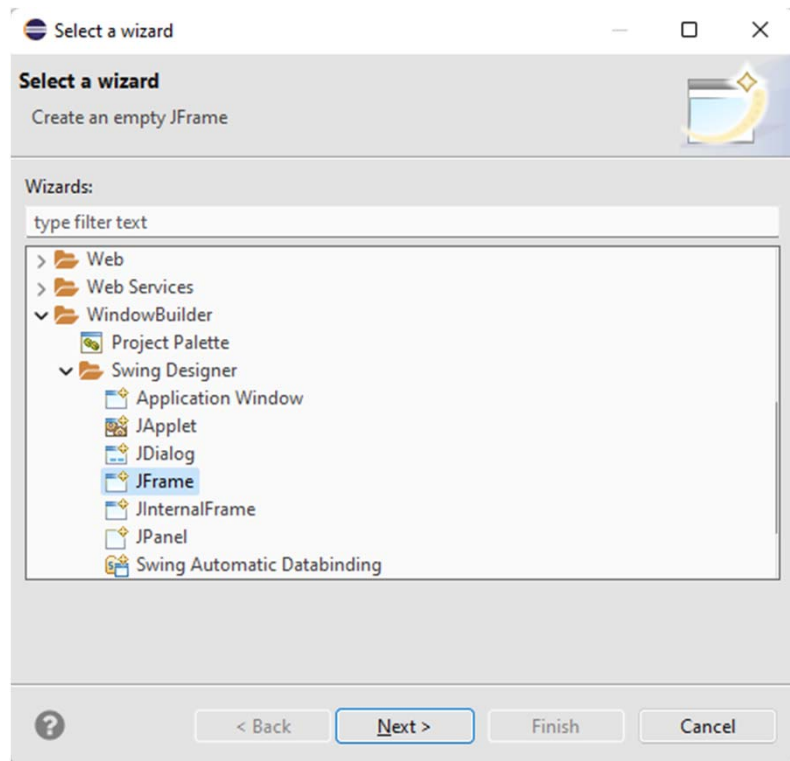
WindowBuilder Nightly Build Nightly

Nightly Build! Install if you want to use the latest patches before a release is available. It can be unstable. WindowBuilder is composed of SWT Designer and... [more info](#)

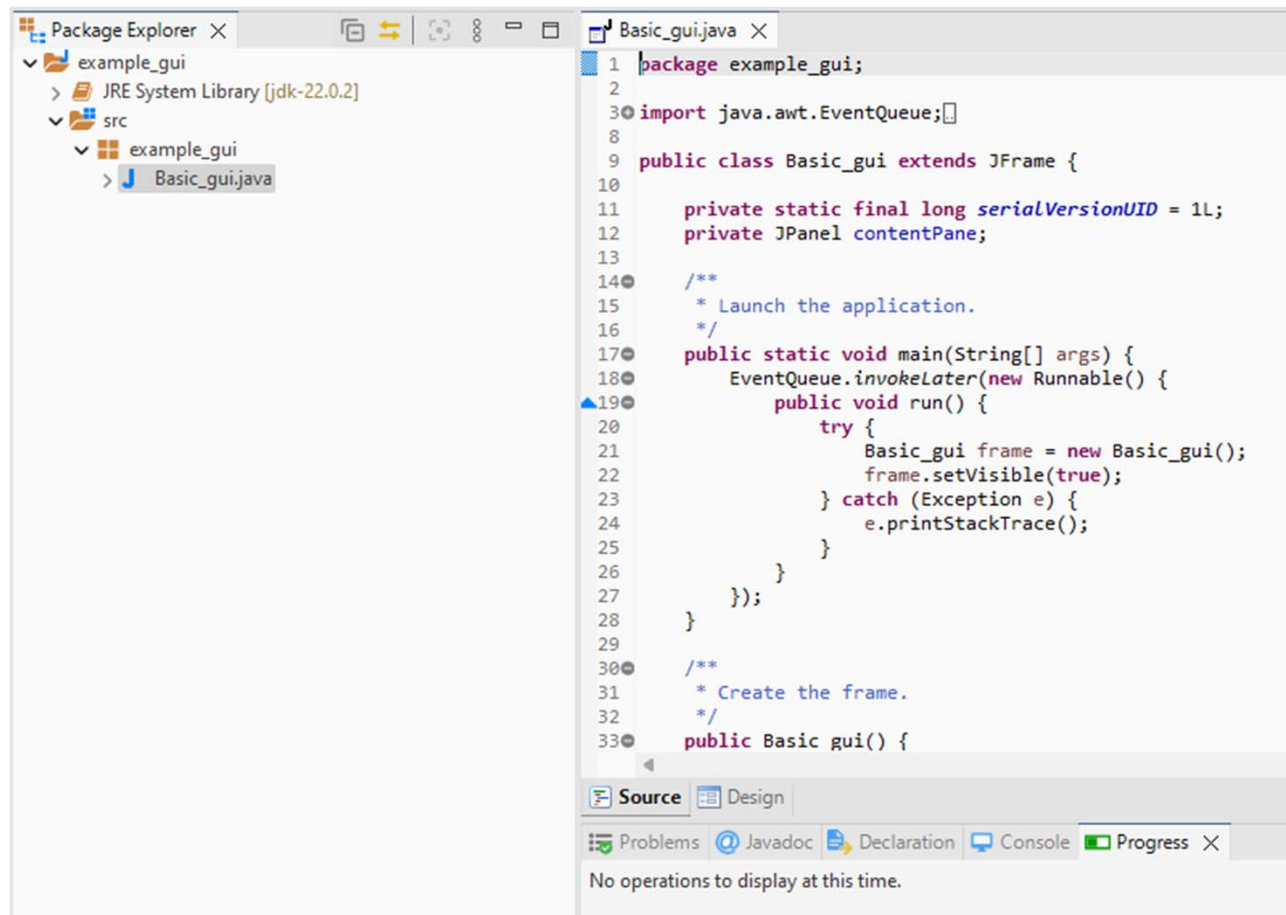
by [Eclipse Foundation](#), EPL
[SWT](#) [swing](#) [wysiwyg](#) [graphical](#) [WindowBuilder](#)

★ 62  Installs: **67.7K** (350 last month) Update ▼

Ejemplo en eclipse

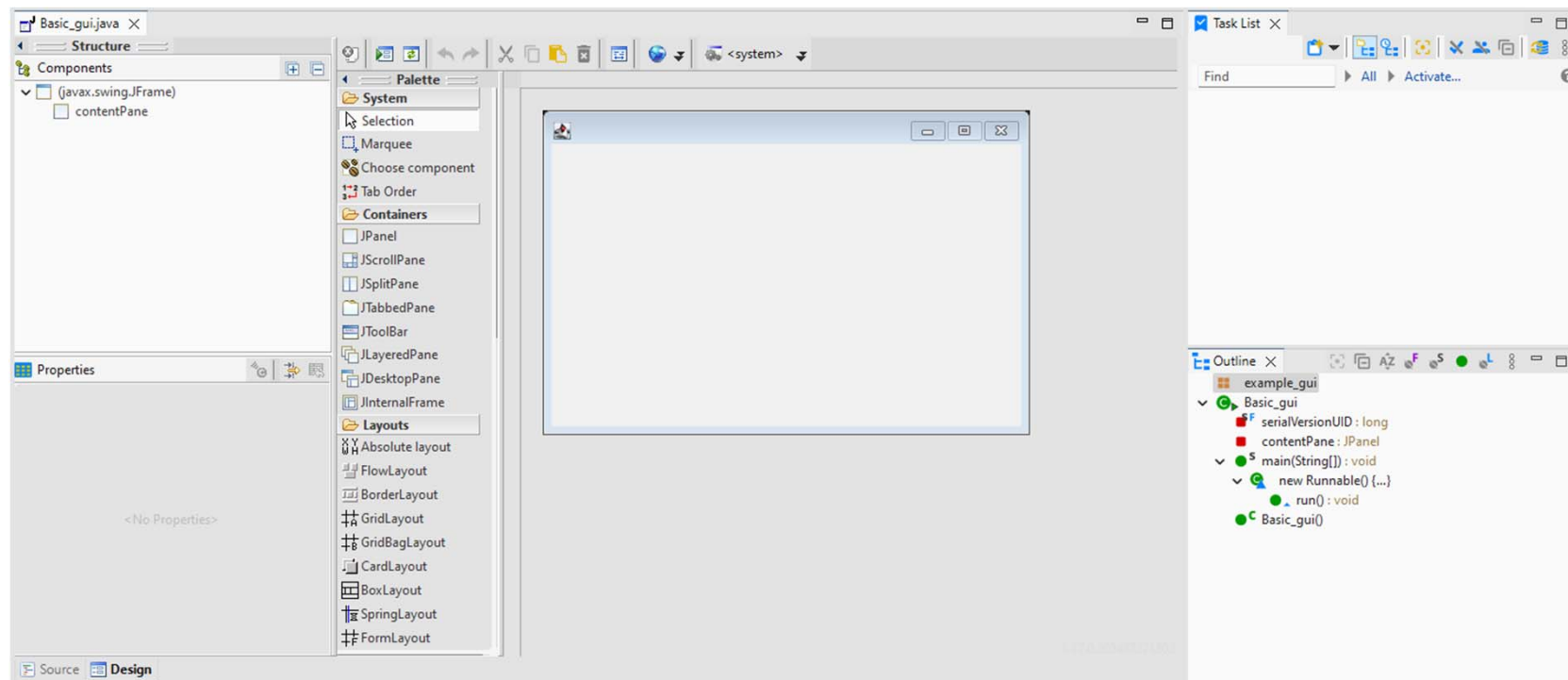


Ejemplo en eclipse

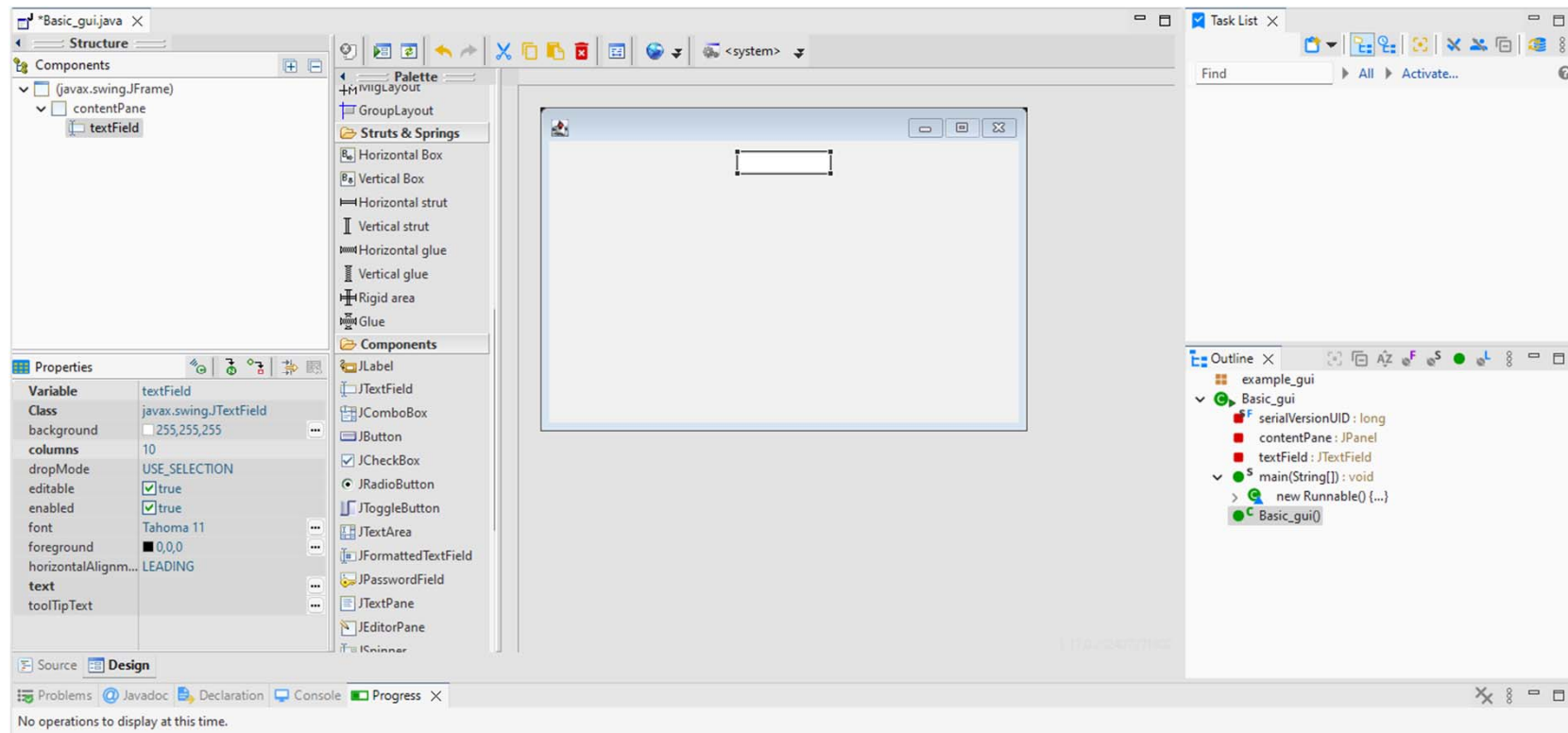


```
1 package example_gui;
2
3 import java.awt.EventQueue;
4
5 public class Basic_gui extends JFrame {
6
7     private static final long serialVersionUID = 1L;
8     private JPanel contentPane;
9
10    /**
11     * Launch the application.
12     */
13    public static void main(String[] args) {
14        EventQueue.invokeLater(new Runnable() {
15            public void run() {
16                try {
17                    Basic_gui frame = new Basic_gui();
18                    frame.setVisible(true);
19                } catch (Exception e) {
20                    e.printStackTrace();
21                }
22            }
23        });
24    }
25
26    /**
27     * Create the frame.
28     */
29    public Basic_gui() {
30    }
```

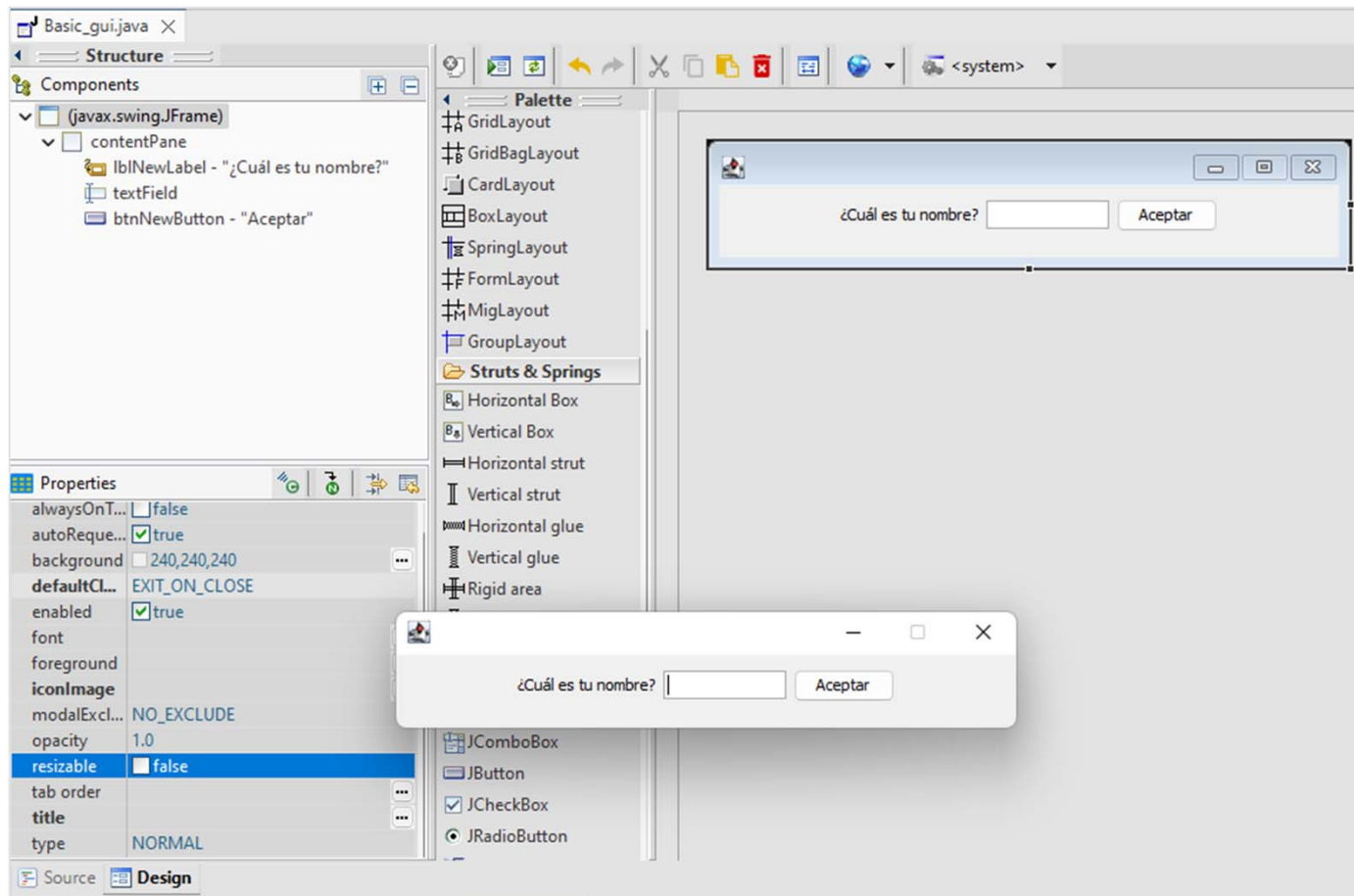
Ejemplo en eclipse



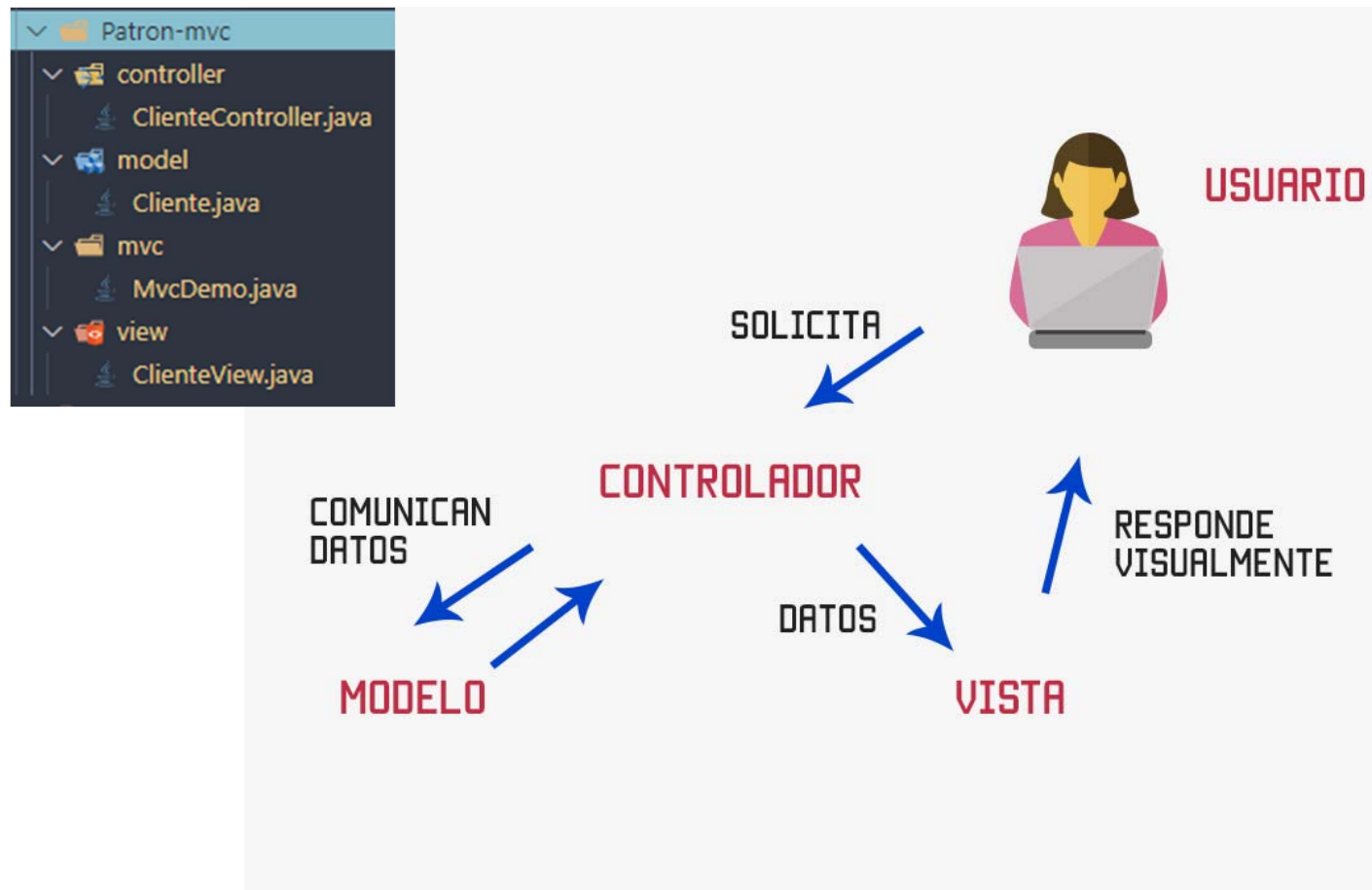
Ejemplo en eclipse



Ejemplo en eclipse



MVC



MVC

The screenshot displays the Eclipse IDE interface for a project named 'patron-mvc'. The Package Explorer on the left shows the project structure: 'example_gui' (containing 'patron-mvc'), 'src' (containing 'controller', 'model', 'mvc', and 'view' packages), and 'view' (containing 'ClienteView.java'). The main editor shows the 'main.java' file with the following code:

```
1 package mvc;
2
3 import controller.ClienteController;
4 import model.Cliente;
5 import view.ClienteView;
6
7 public class main {
8
9     public static void main (String [] args){
10         // objeto vista, y modelo creado con el método estático
11         Cliente modelo= llenarDatosCliente();
12         ClienteView vista= new ClienteView();
13
14         //se crea un objeto controlador y se le pasa el modelo y la vista
15         ClienteController controlador= new ClienteController(modelo, vista);
16
17         // se muestra los datos del cliente
18         controlador.actualizarVista();
19
20         // se actualiza un cliente y se muestra de nuevo los datos
21         controlador.setNombre("Luis");
22         controlador.actualizarVista();
23     }
24
25     //método estático que retorna el cliente con sus datos
26     private static Cliente llenarDatosCliente() {
27         Cliente cliente = new Cliente();
28         cliente.setId(1);
29         cliente.setNombre("Elivar");
30         cliente.setApellido("Largo");
31         return cliente;
32     }
33 }
34
```

The Outline view on the right shows the project structure: 'mvc' (containing 'main'), 'main' (containing 'main(String[]): void'), and 'llenarDatosCliente(): Cliente'.

The Console view at the bottom shows the output of the program:

```
<terminated> main [Java Application] C:\Program Files\OpenJDK\jdk-22.0.2\bin\javaw.exe (Sep 3, 2024, 11:49:39 AM - 11:49:41 AM) [pid: 13440]
**** DATOS CLIENTE ****
Id: 1
Nombre: Elivar
Apellido: Largo
**** DATOS CLIENTE ****
Id: 1
Nombre: Luis
Apellido: Largo
```

MVC

```
1 package mvc;
2
3 import controller.ClienteController;
4 import model.Cliente;
5 import view.ClienteView;
6
7 public class main {
8
9     public static void main (String [] args){
10         // objeto vista, y modelo creado con el método estático
11         Cliente modelo= llenarDatosCliente();
12         ClienteView vista= new ClienteView();
13
14         //se crea un objeto controlador y se le pasa el modelo y la vista
15         ClienteController controlador= new ClienteController(modelo, vista);
16
17         // se muestra los datos del cliente
18         controlador.actualizarVista();
19
20         // se actualiza un cliente y se muestra de nuevo los datos
21         controlador.setNombre("Luis");
22         controlador.actualizarVista();
23     }
24
25     //método estático que retorna el cliente con sus datos
26     private static Cliente llenarDatosCliente() {
27         Cliente cliente = new Cliente();
28         cliente.setId(1);
29         cliente.setNombre("Elivar");
30         cliente.setApellido("Largo");
31         return cliente;
32     }
33 }
34
```

```
1 package model;
2
3 public class Cliente {
4     private int id;
5     private String nombre;
6     private String apellido;
7
8
9     public Cliente() {
10    }
11
12     public int getId() {
13         return id;
14     }
15     public void setId(int id) {
16         this.id = id;
17     }
18
19     public String getNombre() {
20         return nombre;
21     }
22     public void setNombre(String nombre) {
23         this.nombre = nombre;
24     }
25
26     public String getApellido() {
27         return apellido;
28     }
29     public void setApellido(String apellido) {
30         this.apellido = apellido;
31     }
32 }
```

MVC

```
1 package controller;
2
3 import model.Cliente;
4 import view.ClienteView;
5
6 public class ClienteController {
7     //objetos vista y modelo
8     private ClienteView vista;
9     private Cliente modelo;
10
11     //constructor para inicializar el modelo y la vista
12     public ClienteController(Cliente modelo, ClienteView vista) {
13         this.modelo = modelo;
14         this.vista = vista;
15     }
16
17     //getters y setters para el modelo
18     public int getId() {
19         return modelo.getId();
20     }
21     public void setId(int id) {
22         this.modelo.setId(id);
23     }
24     public String getNombre() {
25         return modelo.getNombre();
26     }
27     public void setNombre(String nombre) {
28         this.modelo.setNombre(nombre);
29     }
30     public String getApellido() {
31         return modelo.getApellido();
32     }
33     public void setApellido(String apellido) {
34         this.modelo.setApellido(apellido);
35     }
36
37     //pasa el modelo a la vista para presentar los datos
38     public void actualizarVista() {
39         vista.imprimirDatosCliente(modelo.getId(), modelo.getNombre(), modelo.getApellido());
40     }
41 }
```

```
1 package view;
2
3 public class ClienteView {
4     public void imprimirDatosCliente(int id, String nombre, String apellido) {
5         System.out.println("**** DATOS CLIENTE ****");
6         System.out.println("Id: "+id);
7         System.out.println("Nombre: "+nombre);
8         System.out.println("Apellido: "+apellido);
9     }
10 }
11
```

```
**** DATOS CLIENTE ****
Id: 1
Nombre: Elivar
Apellido: Largo
**** DATOS CLIENTE ****
Id: 1
Nombre: Luis
Apellido: Largo
```


MVC

```
1 package view;
2
3 import java.awt.EventQueue;
4 import javax.swing.JFrame;
5 import javax.swing.JLabel;
6 import javax.swing.JTextField;
7 import java.awt.GridLayout;
8 import java.awt.event.ActionEvent;
9 import java.awt.event.ActionListener;
10 import javax.swing.JButton;
11
12 public class viewGUI extends JFrame implements ActionListener{
13
14     private static final long serialVersionUID = 1L;
15     private JTextField textField_Id;
16     private JTextField textField_nombre;
17     private JTextField textField_Apellido;
18     private JButton btnNewButton;
19
20     /**
21      * Launch the application.
22      */
23     public static void main(String[] args) {
24         EventQueue.invokeLater(new Runnable() {
25             public void run() {
26                 try {
27                     viewGUI frame = new viewGUI();
28                     frame.setVisible(true);
29                 } catch (Exception e) {
30                     e.printStackTrace();
31                 }
32             }
33         });
34     }
35 }
```

```
36 /**
37  * Create the frame.
38  */
39 public viewGUI() {
40     setTitle("Datos del Cliente");
41     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
42     setBounds(100, 100, 360, 150);
43     getContentPane().setLayout(new GridLayout(4, 2));
44
45     JLabel label_Id = new JLabel("Id");
46     getContentPane().add(label_Id);
47
48     textField_Id = new JTextField();
49     getContentPane().add(textField_Id);
50     textField_Id.setColumns(10);
51
52     JLabel label_Nombre = new JLabel("Nombre");
53     getContentPane().add(label_Nombre);
54
55     textField_nombre = new JTextField();
56     getContentPane().add(textField_nombre);
57     textField_nombre.setColumns(10);
58
59     JLabel label_Apellido = new JLabel("Apellido");
60     getContentPane().add(label_Apellido);
61
62     textField_Apellido = new JTextField();
63     getContentPane().add(textField_Apellido);
64     textField_Apellido.setColumns(10);
65
66     btnNewButton = new JButton("Cerrar");
67     btnNewButton.addActionListener(this);
68     getContentPane().add(btnNewButton);
69 }
70
71 @Override
72 public void actionPerformed(ActionEvent e) {
73     if (e.getSource()==btnNewButton) {
74         this.dispose();
75     }
76 }
77
78 public void imprimirDatosCliente(int id,String nombre, String apellido) {
79     textField_Id.setText(Integer.toString(id));
80     textField_nombre.setText(nombre);
81     textField_Apellido.setText(apellido);
82 }
83
84 }
85 }
```


MVC

```
1 package mvc;
2
3 import controller.ClienteController;
4 import model.Cliente;
5 import view.ClienteView;
6 import view.viewGUI;
7
8 public class main {
9
10     public static void main (String [] args){
11         // objeto vista, y modelo creado con el m@todo est@tico
12         Cliente modelo= llenarDatosCliente();
13         //ClienteView vista= new ClienteView();
14         viewGUI vista= new viewGUI();
15
16         //se crea un objeto controlador y se le pasa el modelo y la vista
17         ClienteController controlador= new ClienteController(modelo, vista);
18
19         // se muestra los datos del cliente
20         //controlador.actualizarVista();
21         controlador.actualizarVistaGUI();
22
23         // se actualiza un cliente y se muestra de nuevo los datos
24         controlador.setNombre("Luis");
25         //controlador.actualizarVista();
26         controlador.actualizarVistaGUI();
27         vista.setVisible(true);
28     }
29
30     //método estático que retorna el cliente con sus datos
31     private static Cliente llenarDatosCliente() {
32         Cliente cliente = new Cliente();
33         cliente.setId(1);
34         cliente.setNombre("Elivar");
35         cliente.setApellido("Largo");
36         return cliente;
37     }
38 }
39
```



Datos del Cliente	
Id	1
Nombre	Luis
Apellido	Largo
Cerrar	

```
1 package controller;
2
3 import model.Cliente;
4 import view.ClienteView;
5 import view.viewGUI;
6
7 public class ClienteController {
8     //objetos vista y modelo
9     private ClienteView vista;
10    private Cliente modelo;
11    private viewGUI vista_gui;
12
13    //constructor para inicializar el modelo y la vista
14    public ClienteController(Cliente modelo, ClienteView vista) {
15        this.modelo = modelo;
16        this.vista = vista;
17    }
18
19    public ClienteController(Cliente modelo, viewGUI vista) {
20        this.modelo = modelo;
21        this.vista_gui = vista;
22    }
23
24    //getters y setters para el modelo
25    public int getId() {
26        return modelo.getId();
27    }
28    public void setId(int id) {
29        this.modelo.setId(id);
30    }
31    public String getNombre() {
32        return modelo.getNombre();
33    }
34    public void setNombre(String nombre) {
35        this.modelo.setNombre(nombre);
36    }
37    public String getApellido() {
38        return modelo.getApellido();
39    }
40    public void setApellido(String apellido) {
41        this.modelo.setApellido(apellido);
42    }
43
44    //pasa el modelo a la vista para presentar los datos
45    public void actualizarVista() {
46        vista.imprimirDatosCliente(modelo.getId(), modelo.getNombre(), modelo.getApellido());
47    }
48
49    //pasa el modelo a la vista para presentar los datos
50    public void actualizarVistaGUI() {
51        vista_gui.imprimirDatosCliente(modelo.getId(), modelo.getNombre(), modelo.getApellido());
52    }
53 }
54
55
```

Referencias

- <https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html>
- <https://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html>