

TALLER DE LENGUAJE DE PROGRAMACIÓN I

Entorno de CONSOLA y GRÁFICOS

UNIDAD 4

Diseño y construcción de GUI

JAVA

OBJETIVO

- Historia del GUI (*Graphical User Interface*)
- Paquete java.awt – javax.swing
- Layout (Lay = Poner – Layout = Diseñor - Gestor de diseño)
- Contenedores
- Definición y Características de un Frame
 - Crear Contenedores.
 - `setVisible(true/false);`
 - `setSize(x,y); / setBounds(x,y,ancho,alto);`
 - `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
 - `setIconImage(Image image);`
 - `setTitle(String texto);`
 - `setResizable(boolean resizable);`
 - `setExtendedState(Frame.MAXIMIZED_BOTH);`
 - Escribir en el Frame
 - Componentes.
 - Crear Lienzo
 - JTextField
 - JButton
 - JOptionPane.

Historia

Las Internet Foundation Classes (IFC) eran una **biblioteca gráfica** para el lenguaje de programación Java desarrollada originalmente por Netscape y que se publicó en 1996.

Desde sus inicios el entorno Java ya contaba con una biblioteca de componentes gráficos conocida como AWT (**Abstract Window Toolkit**). Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo.

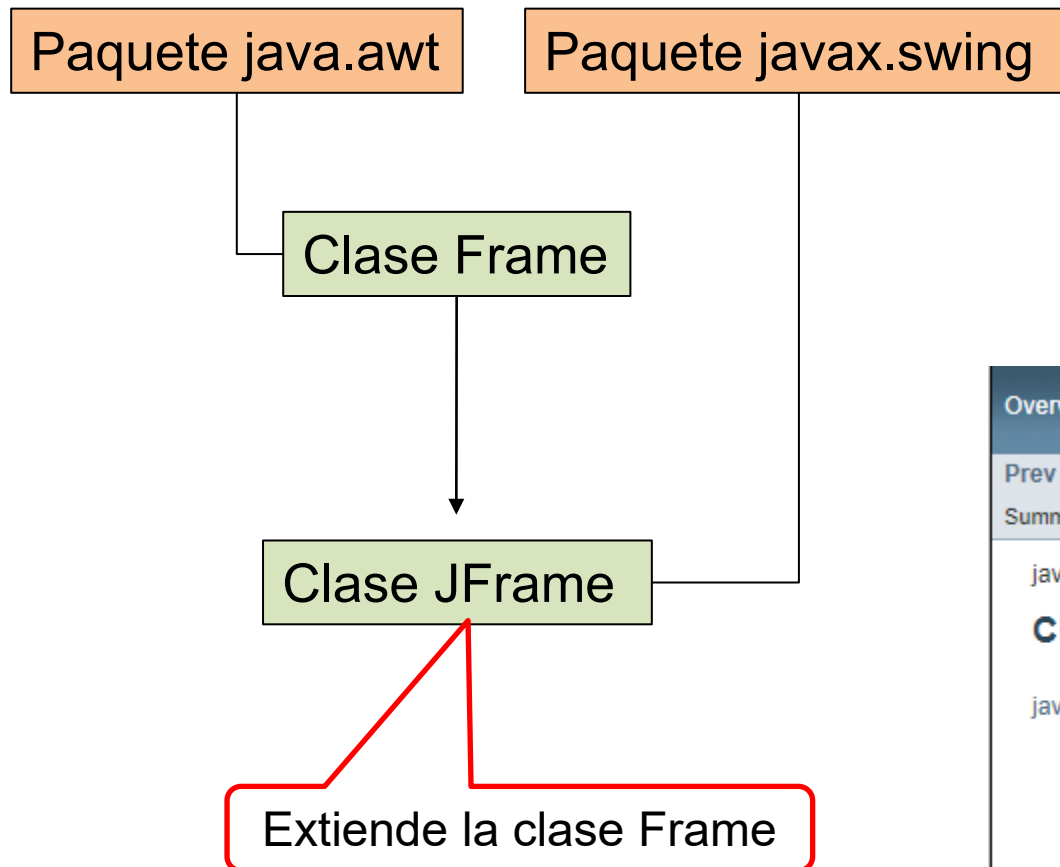
Entonces una aplicación Java corriendo en un sistema operativo específico usaría el botón estándar de ese sistema operativo, que al depender fuertemente de los componentes nativos del sistema operativo.

El comportamiento de los controles varía mucho de sistema a sistema y se vuelve muy difícil construir aplicaciones portables. Fue por esto que el eslogan de Java "Escríbelo una vez, ejecútelo en todos lados" se vio afectado en su implementación.

En cambio, los componentes de IFC eran mostrados y controlados directamente por código Java independiente de la plataforma. De dichos componentes se dice con frecuencia que son componentes ligeros, dado que no requieren reservar recursos nativos del sistema de ventanas del sistema operativo. Además, al estar enteramente desarrollado en Java, aumenta su portabilidad asegurando un comportamiento idéntico en diferentes plataformas.

En 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras tecnologías de las Java Foundation Classes. Además de los componentes ligeros suministrados originalmente por la IFC, Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma. También contiene un conjunto de herramientas que permiten crear una interfaz atractiva para los usuarios.

Paquete java.awt – javax.swing



Overview	Package	Class	Use	Tree	Deprecated	Index	Help
Prev Class	Next Class	Frames	No Frames				
Summary: Nested Field Constr Method		Detail: Field Constr Method					

javax.swing

Class JFrame

java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Window
java.awt.Frame
javax.swing.JFrame

Layout (Lay = Poner – Layout = Diseñor - Gestor de diseño)

Layout

Es la clase que permite administrar el contenido de un contenedor

Los tipos de Layout más comunes son:

BorderLayout → Por defecto del los JFrame

FlowLayout → Por defecto del los JPanel

BoxLayout

GridLayout

CardLayout

Null → Elimina el Layout por defecto

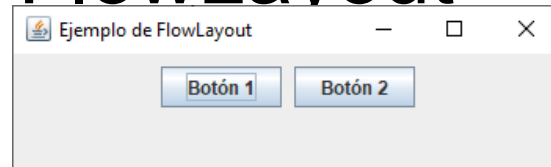
Layout (Lay = Poner – Layout = Diseñor - Gestor de diseño)

Layout

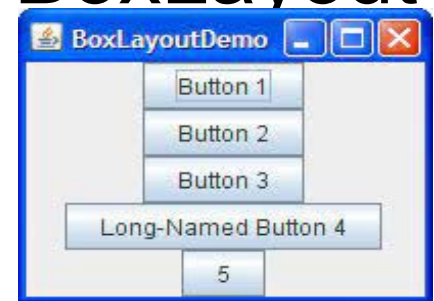
BorderLayout



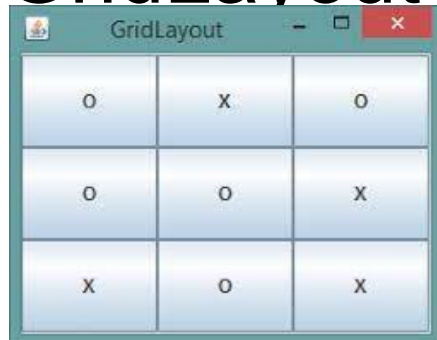
FlowLayout



BoxLayout



GridLayout



CardLayout



Contenedor

CONTENEDOR

Es la clase abstracta **Container** que hereda la clase Component, la función es agrupar objetos gráficos.

Existen varios tipos de contenedores:

- Panel
- Applet
- ScrollPane
- Window
- FileDialog
- Frame

Definición y Características de un Frame

Frame - Marco - Ventana

Frame - Marco - Ventana es un objeto tipo **CONTENEDOR** de objetos gráficos con los que podrá interactuar el usuario con la aplicación. Los Frame poseen la característica típicas de una ventana de programación como minimizar, cerrar, maximizar y poder moverla, etc.

Características:

- Al definir un Frame, **este es no visible**, se debe invocar el método setVisible de la clase JFrame para cambiar el estado y que se presente.
- El tamaño inicial de un Frame **es de 0x0 pixeles**, es necesario invocar el método setSize para modificar el tamaño del Frame.
- Se debe definir la acción al momento de cerrar un Frame

Crear Contenedor (Frame)

- **Método de la clase `java.awt.Window`**

- `setVisible(true/false);`
 - `setSize(x,y);`
 - `setBounds(x,y,ancho,alto);`
- Permite que el Frame sea visible.
- Permite definir el tamaño del Frame

- **Metodos de la clase `JFrame`**

- `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
 - `setIconImage(Image image);`
- Definir la acción al cerrar el Frame
- Definir el icono del Frame

- **Metodos de la clase `java.awt.Frame`**

- `setTitle(String texto);`
 - `setResizable(boolean resizable);`
 - `setExtendedState(Frame.MAXIMIZED_BOTH);`
- Permite definir un título al Frame
- Permite redimensionar el Frame
- Permite ampliar el Frame

Crear Contenedor (Frame)

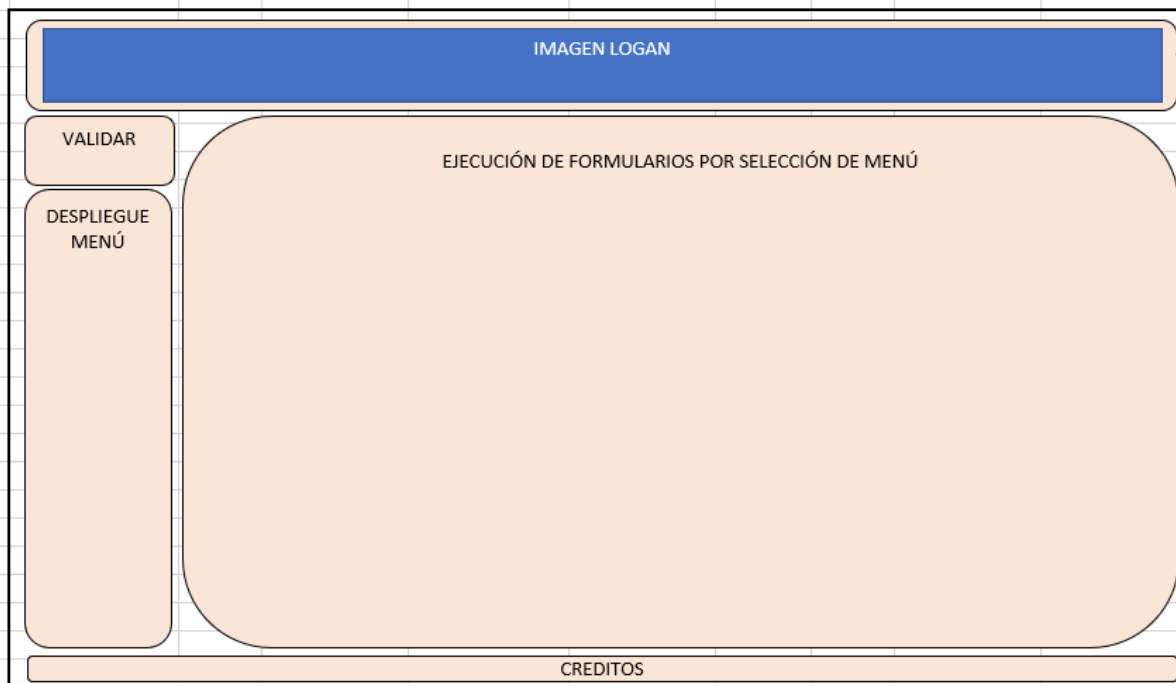
Crear **CONTENEDOR** (JFrame)

- Planeación del diseño
- Instalación del WindowBuilder
(Herramienta de apoyo al diseño)
- Codificación

Planeación General Contenedor - JFrame

Codigo	Nombre	Dependencia	Formulario	Estado	Orden
1	Maestras	0		A	1
2	Menú	4	Vmenu	A	1
3	Sexos	4	VSexos	A	2
4	Básicas	1		A	1
5	Operaciones	0		A	2
6	Empleados	5	VEmpleados	A	1

Estado= A -- Activo
I -- Inactivo



Fondo color: 195,195,195

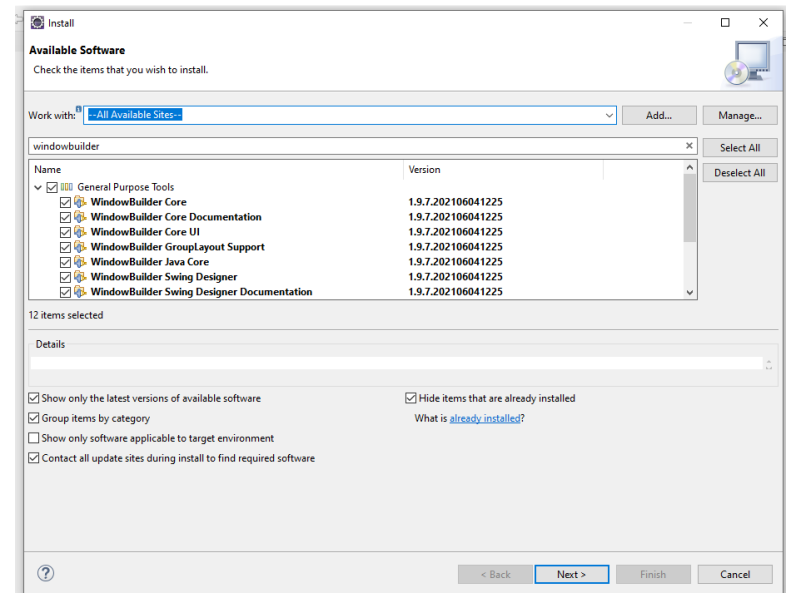
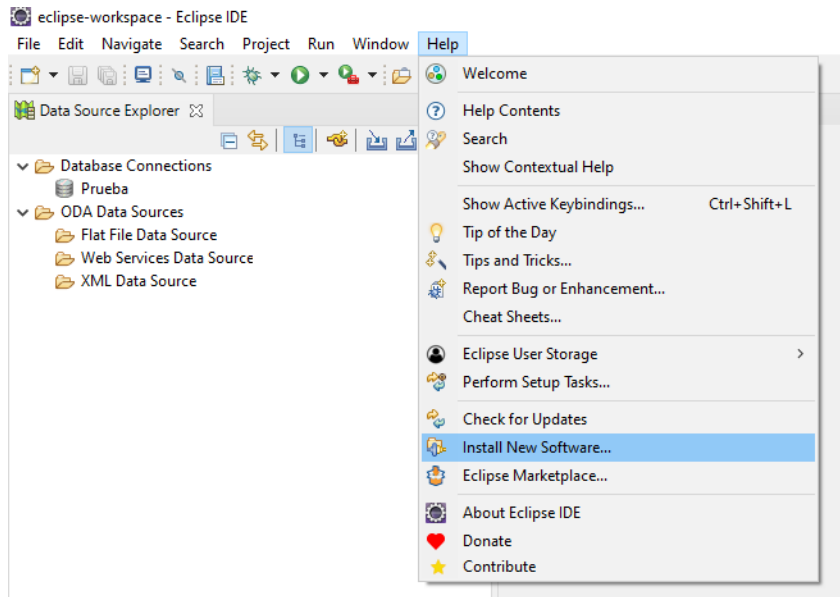
Panel Superior

Panel Central

Panel Inferior

WindowBuilder

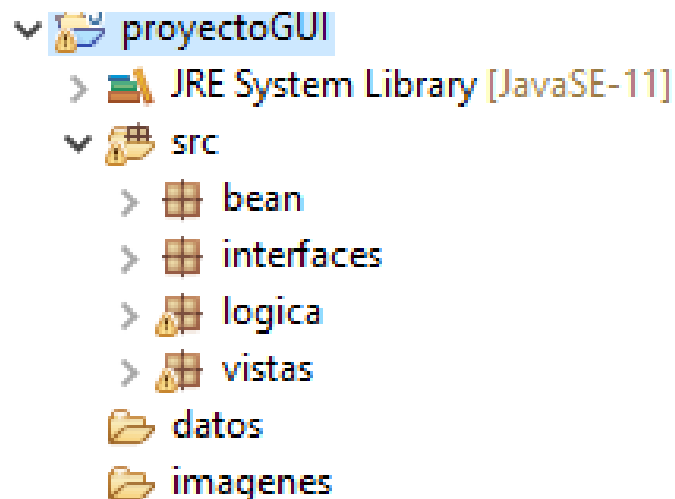
Instalar la librería WindowBuilder



Proyecto

Se desarrollará una aplicación que permita almacenar y desplegar un Menú de navegación.

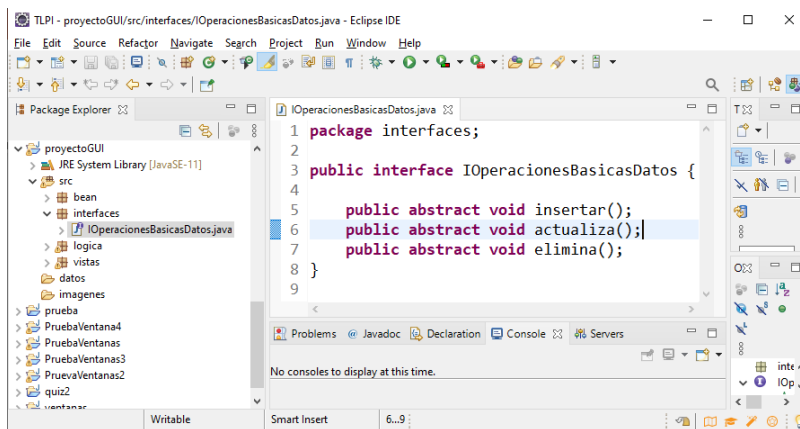
Crear proyecto llamado proyectoGUI con los paquetes: bean, logica, interfaces y vista.



Codificación

Siguiente Paso

- Crear la clase tipo interface `IOperacionesBasicasDatos` en el paquete `interfaces`, donde se definan los métodos abstractos `insertar`, `actualizar` y `eliminar`.



```
package interfaces;  
  
public interface IOperacionesBasicasDatos  
{  
  
    public abstract void insertar();  
    public abstract void actualiza();  
    public abstract void elimina();  
}
```

Codificación

Siguiente Paso

- Crear la clase BMenu dentro del paquete bean, sin método main con los atributos y métodos getter y setter.

```
package bean;

public class BMenu {
    private byte codigo_menu;
    private String nombre_menu;
    private byte dependencia;
    private String formulario;
    private String estado;
    private int orden;

    public BMenu()
    {
        this.codigo_menu = 0;
        this.nombre_menu = "";
        this.dependencia = 0;
        this.formulario = "";
        this.estado = "I";
        this.orden = 1;
    }
}
```

Adicionar los métodos
getters y setters de los
atributos

Codificación

Siguiente Paso

- Crear la clase LMenu dentro del paquete lógica, extienda la clase BMenu e implemente la interface IOperacionesBasicasDatos.
- Crear una Folder llamado datos dentro del proyecto.

```
package logica;

import bean.BMenu;
import interfaces.IOperacionesBasicasDatos;

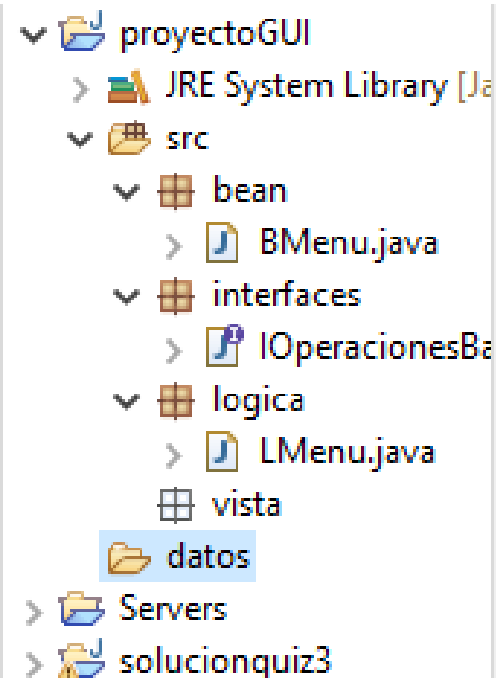
public class LMenu extends BMenu implements
IOperacionesBasicasDatos
{

    @Override
    public void insertar() {
        // TODO Auto-generated method stub
    }

    @Override
    public void actualiza() {
        // TODO Auto-generated method stub
    }

    @Override
    public void elimina() {
        // TODO Auto-generated method stub
    }

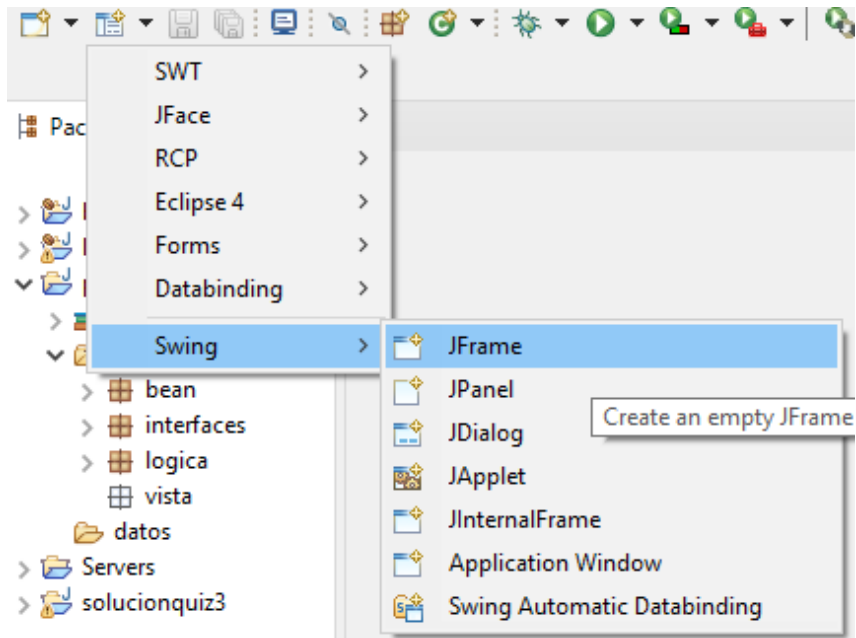
}
```



Crear Contenedor (JFrame)

Siguiente Paso

Crear una clase con nombre Inicio en el paquete vista con método main de tipo JFrame

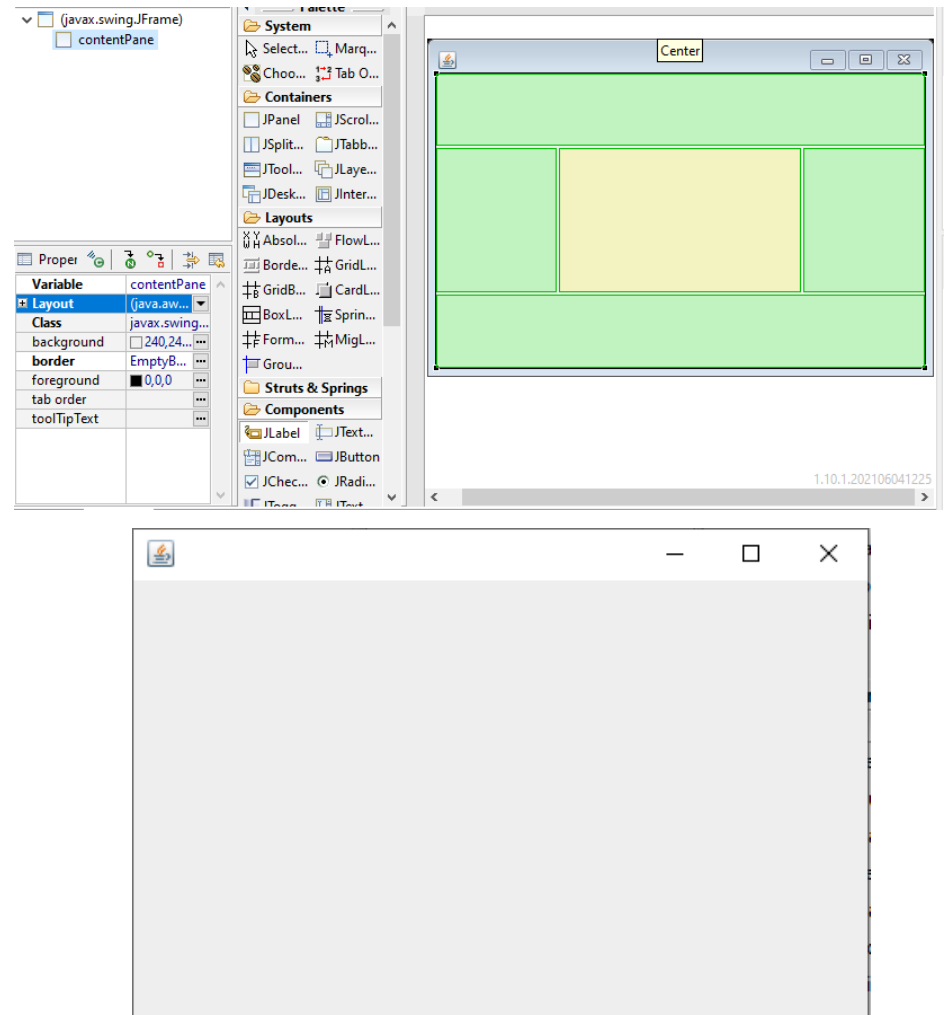


De forma automática se definen los parámetros:

- setVisible(**true**);
- setLocation(x,y);
- setDefaultCloseOperation(JFrame **.EXIT_ON_CLOSE**);

Crear Contenedor (JFrame)

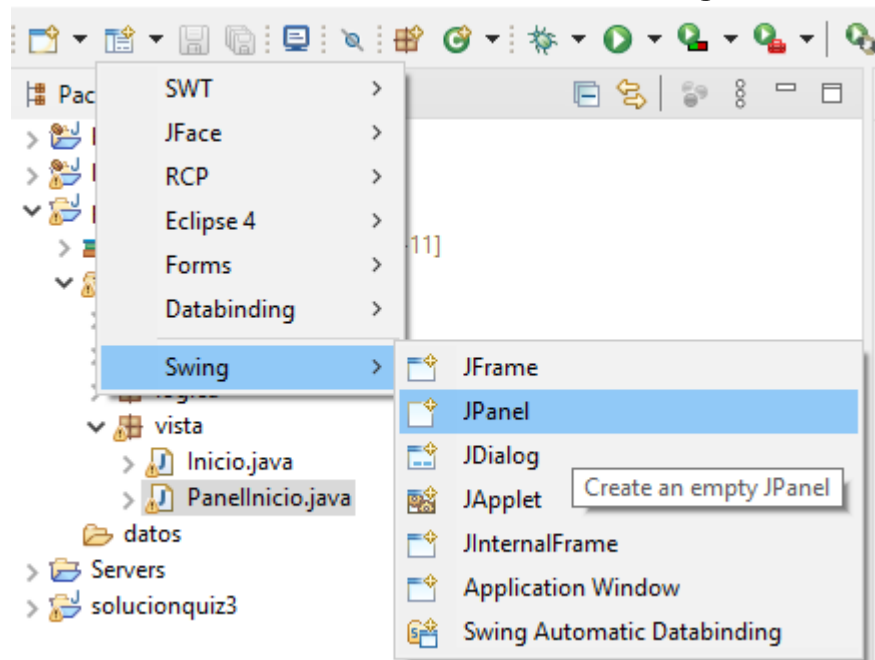
```
public Inicio()  
{  
    setDefaultCloseOperation(J  
Frame.EXIT_ON_CLOSE);  
    setBounds(100, 100, 450,  
300);  
    contentPane = new  
JPanel();  
    contentPane.setBorder(new  
EmptyBorder(5, 5, 5, 5));  
  
    contentPane.setLayout(new  
BorderLayout(0, 0));  
    setContentPane(contentPane  
);  
}
```



Crear Contenedor (JPanel)

Siguiente Paso

Crear una clase con nombre `PanelInicio`, será el contenedor de los objetos según el diseño planeado



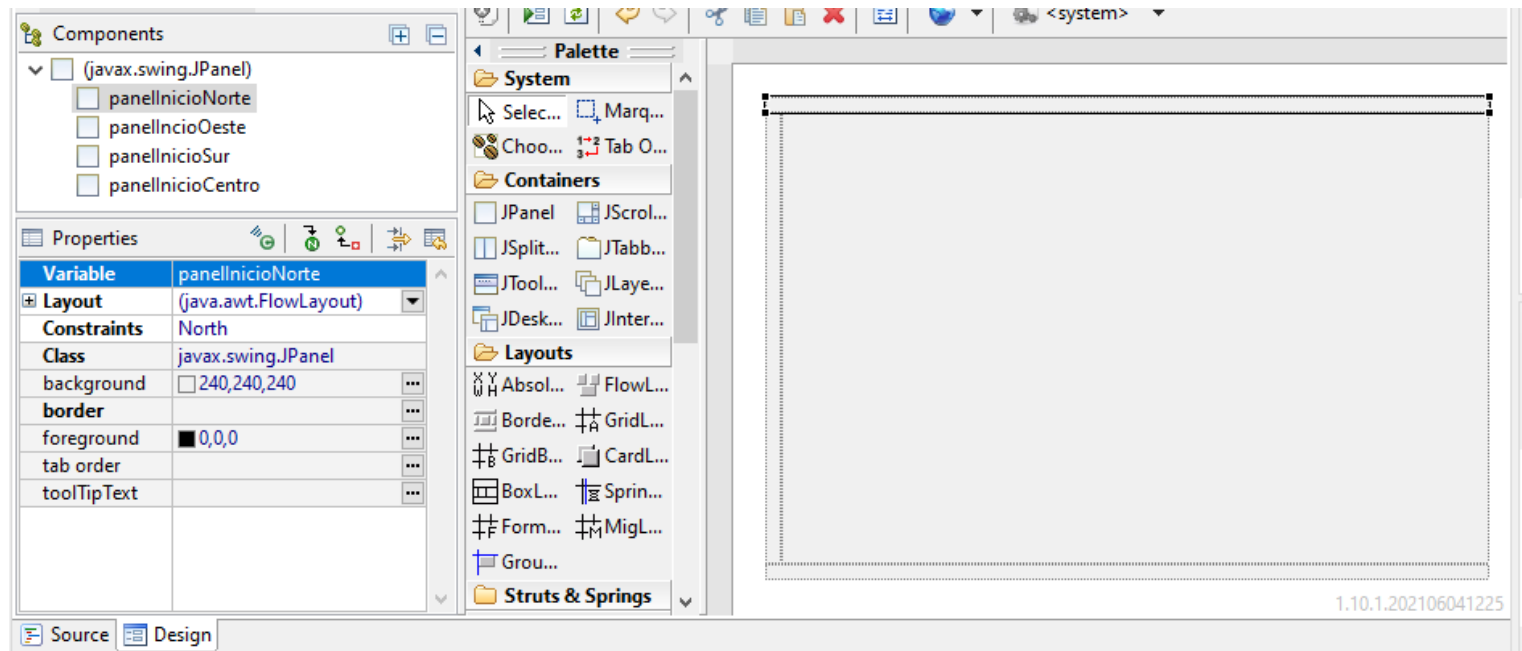
Se adiciona el `PanelInicio` al `JFrame` de la clase `Inicio` en la zona central

```
PanelInicio pinicio = new  
PanelInicio();  
contentPane.add(pinicio,  
BorderLayout.CENTER);
```

Crear Contenedor (JPanel)

Siguiente Paso

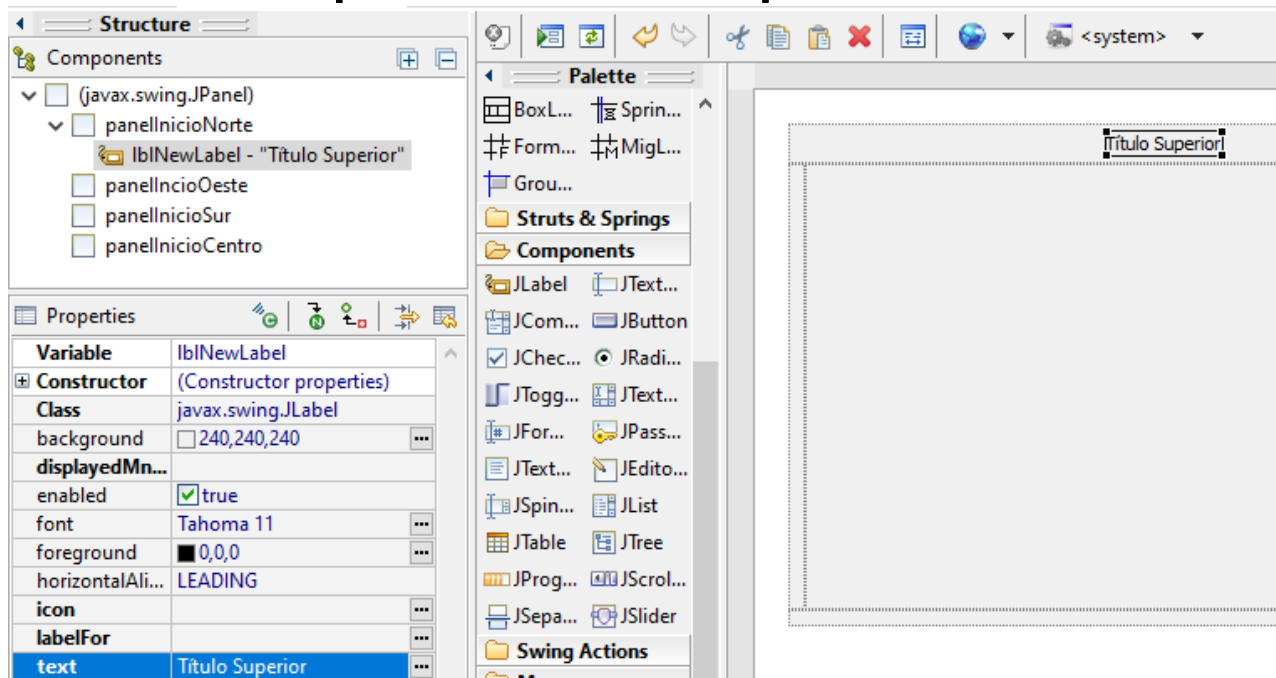
Definir JPanel para la zona Norte, Oeste, Centro y Sur del BorderLayout de PanelInicio



Crear Contenedor (JPanel)

Siguiente Paso

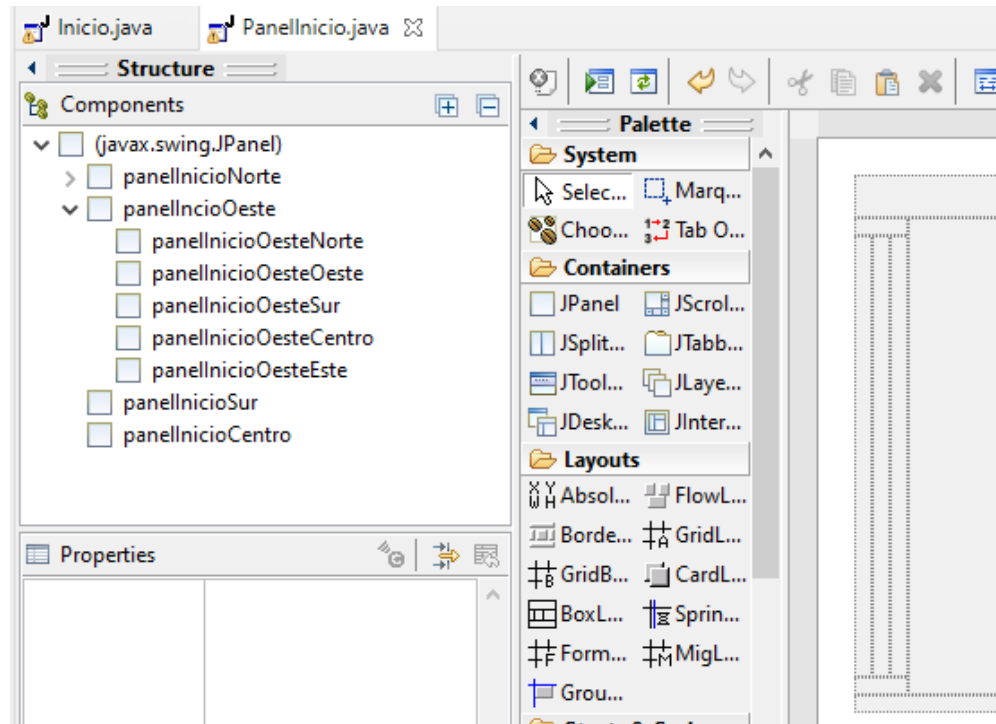
En panelInicioNorte, adicionar un objeto tipo JLabel, cambiar el texto por Título Superior



Crear Contenedor (JPanel)

Siguiente Paso

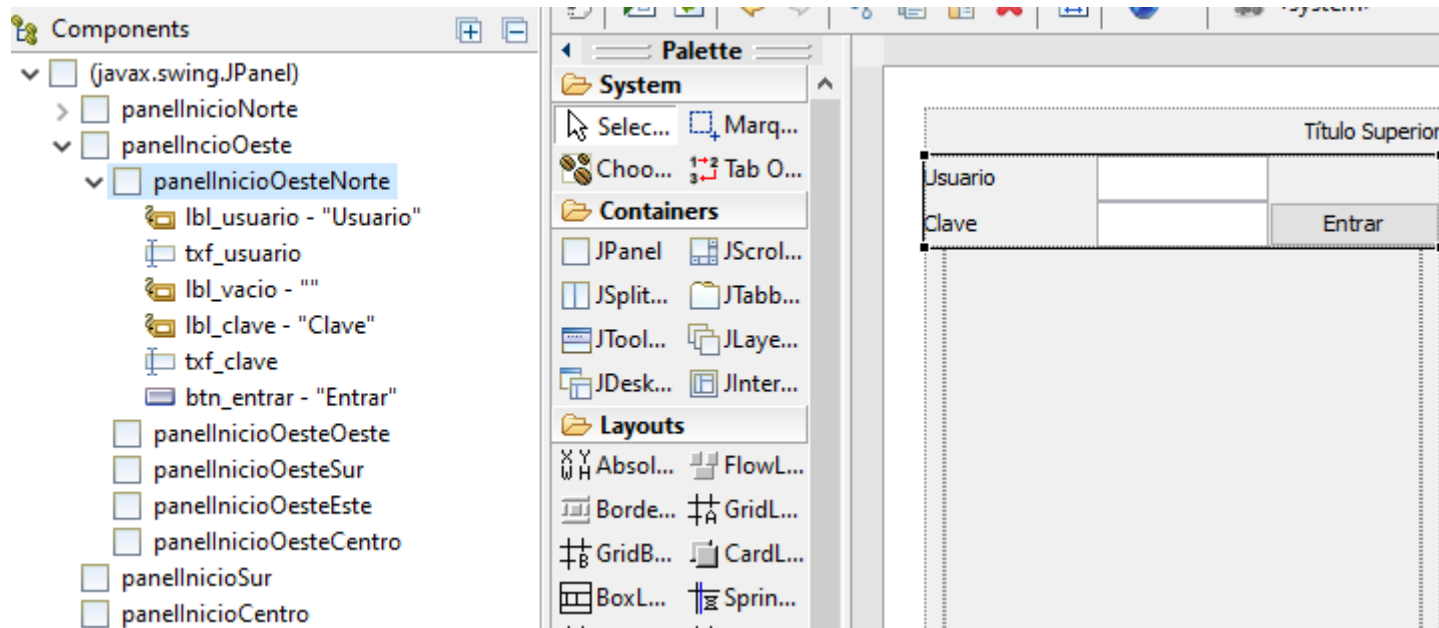
En el panel `panellInicioOeste`, modificar el Layout por defecto a `BorderLayout`, para cada zona adicionar un `JPanel`



Crear Contenedor (JPanel)

Siguiente Paso

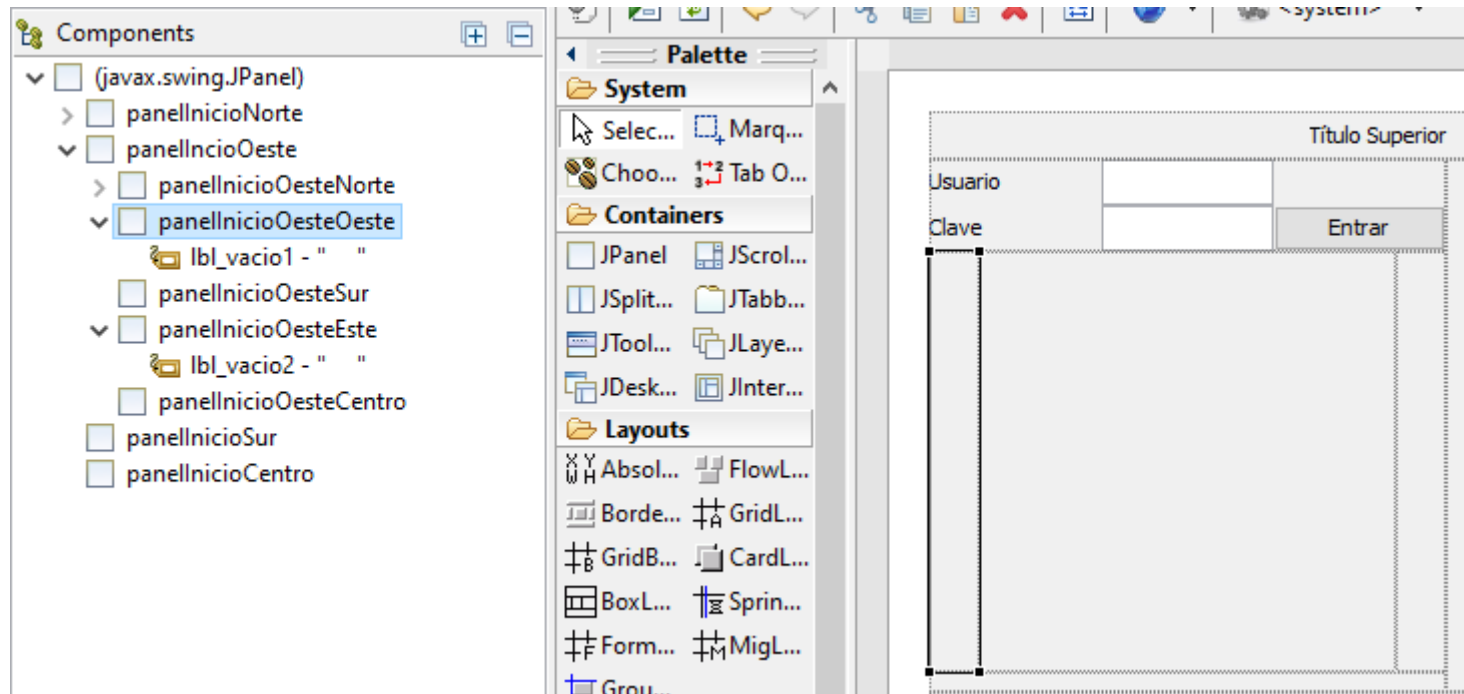
En el panel `panelInicioOesteNorte`, adicionar tres objetos `JLabel`, dos `JTextFiel` y un `Jbottun`, modificar el Layout por defecto a `GridLayout`, especificar una matriz de 2X3



Crear Contenedor (JPanel)

Siguiente Paso

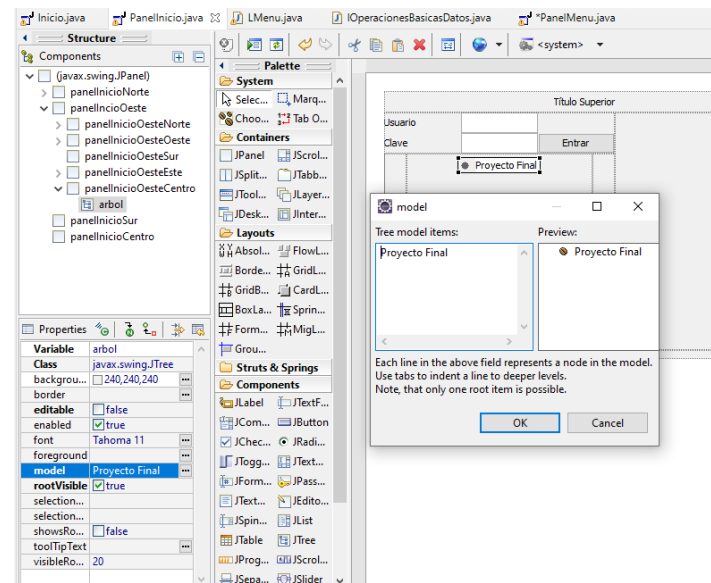
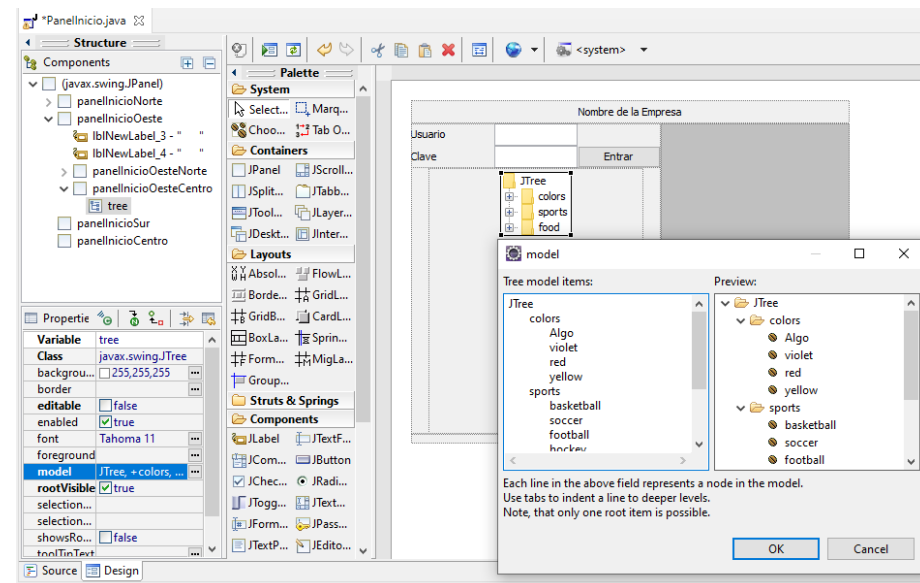
En el panel `panelInicioOesteOeste`, adicionar un objetos `JLabel`, modificar el parámetro `text` con 5 espacios, igual para el `panelInicioOesteEste`



Crear Contenedor (JPanel)

Siguiente Paso

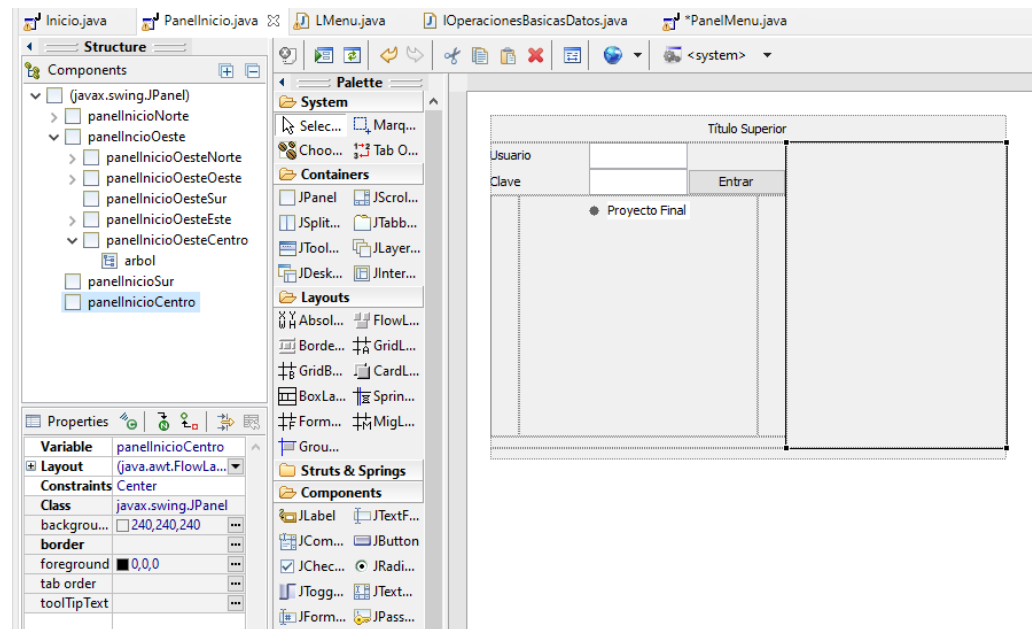
En el panel `panelInicioOesteCentro`, adicionar un objetos `JTree`, por el parámetro modal del `JTree`, eliminar los item y la opción `JTree`, modificar por `Practica Final`



Crear Contenedor (JPanel)

Siguiente Paso

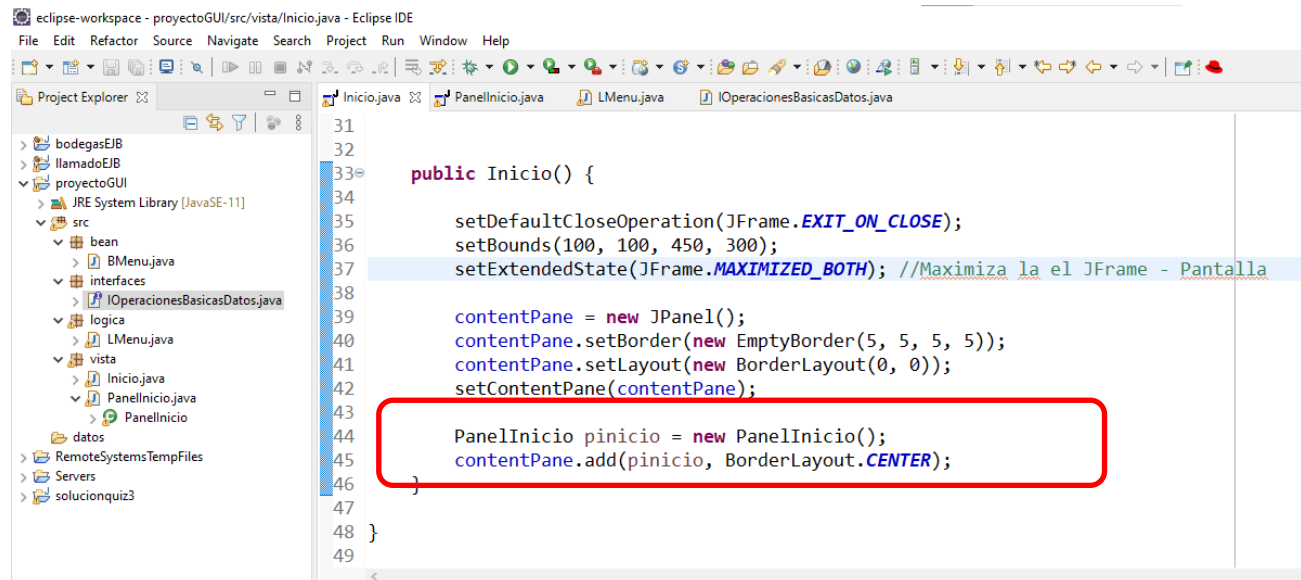
Adicionar un JPanel en la zona centro del JPanel PanelInicio, en este panel se cargaran los demás paneles, según opción seleccionada del menú



Crear Contenedor (JPanel)

Siguiente Paso

En la clase Inicio, se define un objeto de tipo PanelInicio, lo adicionamos a la zona centro del layout por defecto del JFrame



```
31
32
33 public Inicio() {
34
35     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
36     setBounds(100, 100, 450, 300);
37     setExtendedState(JFrame.MAXIMIZED_BOTH); //Maximiza la el JFrame - Pantalla
38
39     contentPane = new JPanel();
40     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
41     contentPane.setLayout(new BorderLayout(0, 0));
42     setContentPane(contentPane);
43
44     PanelInicio pinicio = new PanelInicio();
45     contentPane.add(pinicio, BorderLayout.CENTER);
46 }
47
48 }
49
```

```
PanelInicio
pinicio = new
PanelInicio();
contentPane.add
(pinicio,
BorderLayout.CE
NTER);
```

Crear Contenedor (JPanel)

Siguiente Paso

En la clase LMenu definir el método listarMenu() así:

```
public ArrayList<BMenu> listarMenu(int nivel) {
    String wcadena="";
    int cuantos=0;

    ArrayList<BMenu> listaMenu = new ArrayList();
    try
    {
        File existeArchivo = new File("C:\\Users\\Administrador\\eclipse-workspace\\proyectoGUI\\datos\\datosMenu.dat");
        if (existeArchivo.exists())
        {
            RandomAccessFile archivo = new RandomAccessFile("C:\\Users\\Administrador\\eclipse-workspace\\proyectoGUI\\datos\\datosMenu.dat", "r");
            archivo.seek(0);
            while(archivo.getFilePointer() < archivo.length())
            {
                BMenu datosMenu = new BMenu();
                datosMenu.setCodigoMenu(archivo.readInt());
                datosMenu.setNombreMenu(archivo.readUTF());
                datosMenu.setDependenciaMenu(archivo.readInt());
                datosMenu.setFormularioMenu(archivo.readUTF());
                datosMenu.setEstadoMenu(archivo.readUTF());
                datosMenu.setOrdenMenu(archivo.readInt());

                if(datosMenu.getDependenciaMenu() == nivel)
                {
                    listaMenu.add(datosMenu);
                }
                cuantos++;
            }
            if(cuantos == 0) {
                consecutivoMenu();
            }
            archivo.close();
        }
        else
        {
            consecutivoMenu();
        }
    }
    catch(IOException e){
        System.out.print("aquí " + e.getCause());
        consecutivoMenu();
    }
    return listaMenu;
}
```

Crear Contenedor (JPanel)

Siguiente Paso

En la clase LMenu definir el método listarMenuTodos() así:

```
public ArrayList<BMenu> listarMenuTodos(){
ArrayList<BMenu> listaMenu = new ArrayList();
    try
    {
File existeArchivo = new File("C:\\Users\\Administrador\\eclipse-workspace\\proyectoGUI\\datos\\datosMenu.dat");
if (existeArchivo.exists())
{
RandomAccessFile archivo = new RandomAccessFile("C:\\Users\\Administrador\\eclipse-workspace\\proyectoGUI\\datos\\datosMenu.dat", "r");
archivo.seek(0);
        while(archivo.getFilePointer() < archivo.length())
        {

            BMenu datosMenu = new BMenu();
            datosMenu.setCodigoMenu(archivo.readInt());
            datosMenu.setNombreMenu(archivo.readUTF());
            datosMenu.setDependenciaMenu(archivo.readInt());
            datosMenu.setFormularioMenu(archivo.readUTF());
            datosMenu.setEstadoMenu(archivo.readUTF());
            datosMenu.setOrdenMenu(archivo.readInt());

            listaMenu.add(datosMenu);
        }
        archivo.close();
    }else
    {
consecutivoMenu();
    }
}catch(IOException e)
{
System.out.print("aquí " + e.getCause());
consecutivoMenu();
}
return listaMenu;
}
```

Crear Contenedor (JPanel)

Siguiente Paso

Una vez definió el JTree, se programa el escucha del JTree

```
DefaultMutableTreeNode raiz = new DefaultMutableTreeNode("Proyecto Final");
DefaultTreeModel modelo = new DefaultTreeModel(raiz);

JTree arbol = new JTree();
arbol.addTreeSelectionListener(new javax.swing.event.TreeSelectionListener()
{
    public void valueChanged(javax.swing.event.TreeSelectionEvent evt)
    {
        jTree1ValueChanged(evt);
    }
});

arbol.setModel(modelo);
```

```
public void jTree1ValueChanged( TreeSelectionEvent tse) {
    String node = tse.getNewLeadSelectionPath().getLastPathComponent().toString();

    if( node.equals("Menú") ) {
        //System.out.println(node);
        PanelMenu panelMenu = new PanelMenu();
        add(panelMenu, BorderLayout.CENTER);
    } else if( node.equals("video") ) {
        // play video
    }
}
```

Crear Contenedor (JPanel)

Siguiente Paso

Se define el método privado que escucha la selección en el JTree

```
DefaultMutableTreeNode raiz = new DefaultMutableTreeNode("Proyecto Final");
DefaultTreeModel modelo = new DefaultTreeModel(raiz);

menu = opcionesMenu.listarMenu(0);
menubuscar = opcionesMenu.listarMenuTodos();

JTree arbol = new JTree();

arbol.addTreeSelectionListener(new javax.swing.event.TreeSelectionListener()
{
    public void valueChanged(javax.swing.event.TreeSelectionEvent evt)
    {
        for(int q=0;q< menubuscar.size();q++)
        {
            if(menubuscar.get(q).getNombreMenu().equals(String.valueOf(arbol.getLastSelectedPathComponent())) {
                formulario = menubuscar.get(q).getFormularioMenu();
                jTree1ValueChanged(evt,menubuscar.get(q).getFormularioMenu());
            }
        }
    }
});
```

Crear Contenedor (JPanel)

Siguiente Paso

Se define la clase PanelMenu de tipo JPanel, donde se cargara la información que construye el menú.

Modificar el layout por defecto FlouLayout por BorderLayout

- Zona norte, incluir un Jpanel con layout por defecto
- Zona centro, incluir un Jpanel con layout BorderLayout.
 - Incluir un objeto JScrollPane
 - Incluir un objeto Jtable

Crear Contenedor (JPanel)

Siguiente Paso

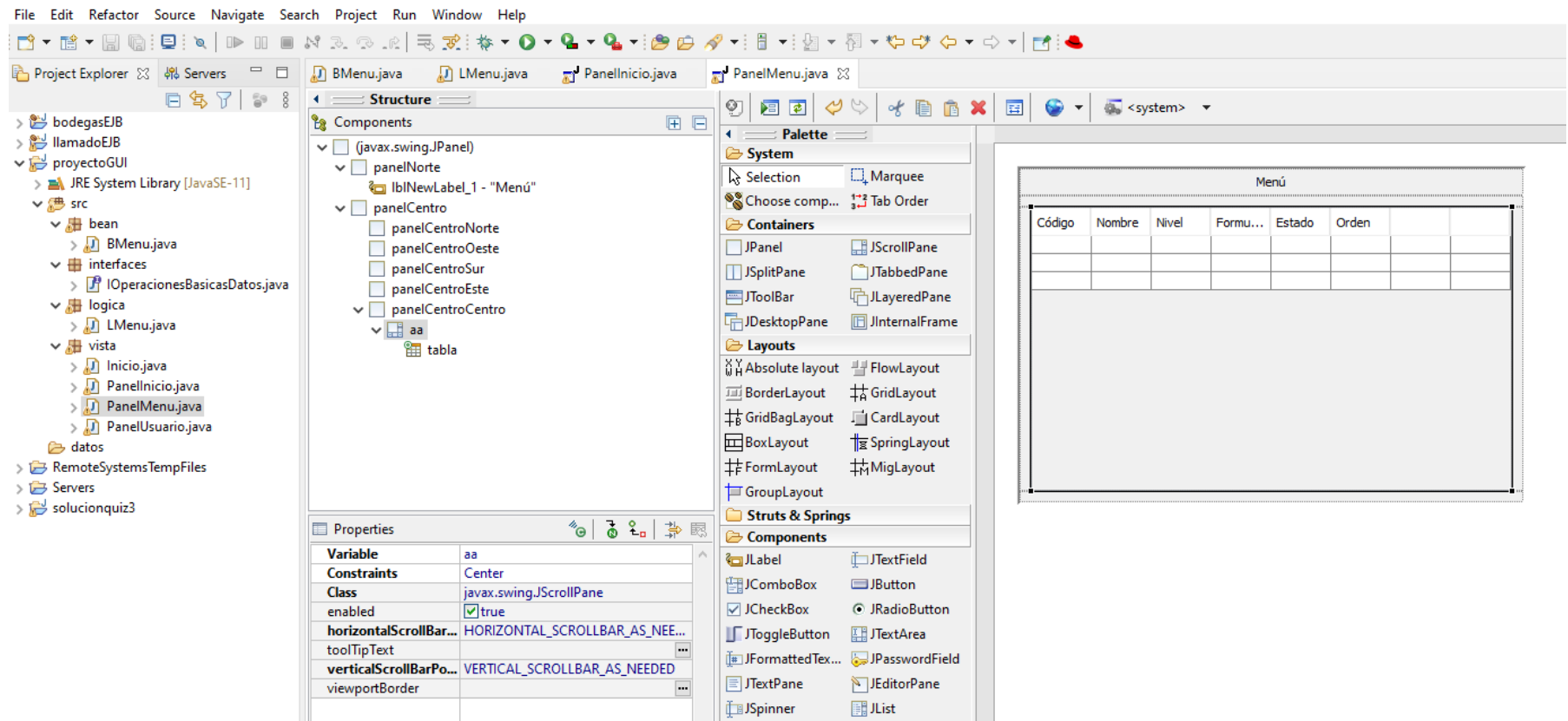
Se define la clase PanelMenu de tipo JPanel, donde se cargara la información que construye el menú.

Modificar el layout por defecto FlouLayout por BorderLayout

- Zona norte, incluir un JPanel con layout por defecto
 - Adicionar un JLabel para el título de la opción.
- Zona centro, incluir un JPanel con layout BorderLayout.
 - Incluir un objeto JScrollPane, en la zona centro
 - Incluir un objeto JTable

Crear Contenedor (JPanel)

Siguiente Paso



Crear Contenedor (JPanel)

Siguiente Paso

Se traen los datos del archivo de datos que almacena la información del menú

```
ArrayList<BMenu> menu = new ArrayList<BMenu>();
LMenu opcionesMenu = new LMenu();
menu = opcionesMenu.listarMenuTodos();

JTable tabla;
DefaultTableModel modelo;
Object[][] datos=new Object[menu.size()][8];
Object [] fila=new Object[3];

String[] nombreColumnas1 = {"Código", "Nombre", "Nivel", "Formulario", "Estado", "Orden", " ", " "};

for(int i=0; i< menu.size();i++)
{
    datos[i][0] = menu.get(i).getCodigoMenu();
    datos[i][1] = menu.get(i).getNombreMenu();
    datos[i][2] = menu.get(i).getDependenciaMenu();
    datos[i][3] = menu.get(i).getFormularioMenu();
    datos[i][4] = menu.get(i).getEstadoMenu();
    datos[i][5] = menu.get(i).getOrdenMenu();
    datos[i][6] = "Modificar";
    datos[i][7] = "Eliminar";
}
modelo=new DefaultTableModel(datos,nombreColumnas1);
tabla=new JTable(modelo);
tabla.addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e) {
        JOptionPane.showMessageDialog(null, "Dio click");
    }
});
panelCentroCentro.setLayout(new BorderLayout(0, 0));

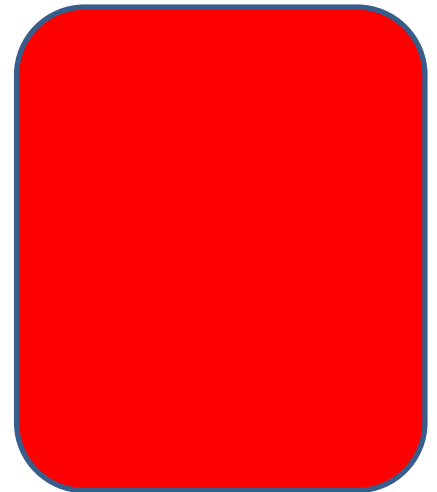
JScrollPane aa = new JScrollPane();
aa.setViewportViewView(tabla);

panelCentroCentro.add(aa);
```

Crear Contenedor (Frame)

Instanciar la clase `VentanaPrincipal` en la clase que contiene el main, para este ejercicio sería `Inicio`

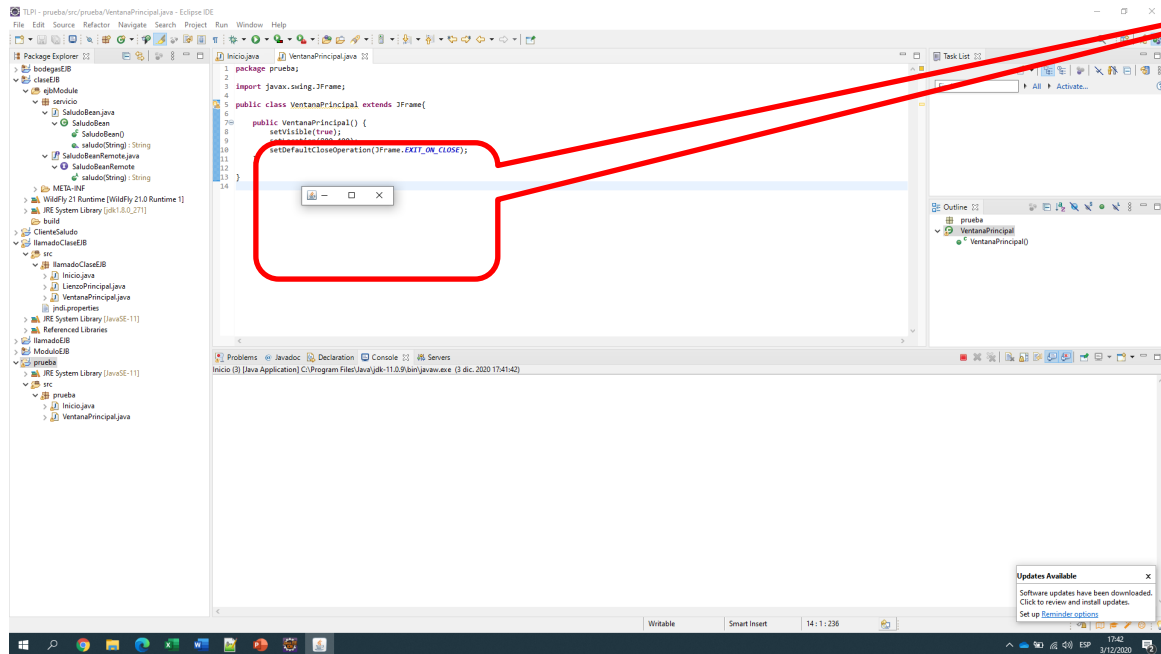
```
package llamadoClaseEJB;  
  
import javax.swing.JFrame;  
  
public class VentanaPrincipal extends JFrame{  
  
    public VentanaPrincipal() {  
    }  
}
```



Si se ejecuta la clase `Inicio`, el resultado es nada.

Crear Contenedor (Frame)

Al ejecutar la clase Inicio, el resultado es:



Se debe invocar el método `setSize(x,y)`

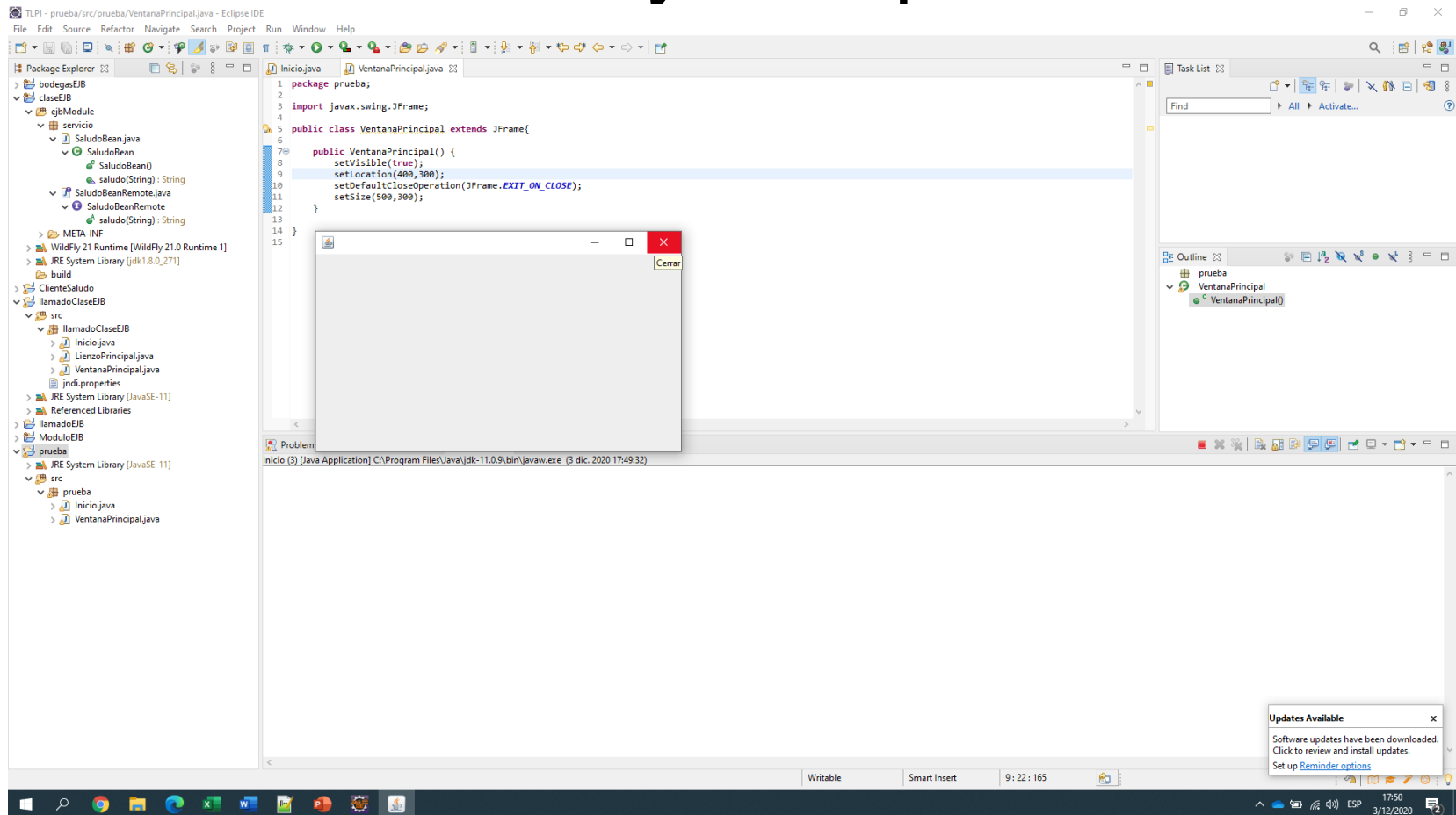
Crear Contenedor (Frame)

Invocar el método setSize(x,y)

```
package llamadoClaseEJB;  
import javax.swing.JFrame;  
public class VentanaPrincipal extends JFrame{  
    public VentanaPrincipal() {  
        setVisible(true);  
        setLocation(800,400);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setSize(500,300);  
    }  
}
```

Crear Contenedor (Frame)

El Frame es visible y manipulable:



Crear Contenedor (Frame)

Algunos métodos que podría ser utilizados

```
package llamadoClaseEJB;

import java.awt.Image;
import java.awt.Toolkit;

import javax.swing.JFrame;

public class VentanaPrincipal extends JFrame{

    public VentanaPrincipal() {
        setVisible(true); //Volver visible el Frame
        //setSize(500,300);
        //setLocation(500,300);

        Toolkit pantalla = Toolkit.getDefaultToolkit(); //Permite conocer el tamaño de la pantalla nativa
        Dimension tamanoPantalla = pantalla.getScreenSize();
        int altoPantalla = tamanoPantalla.height;
        int anchoPantalla = tamanoPantalla.width;
        setBounds(anchoPantalla/4,altoPantalla/4,anchoPantalla/2,altoPantalla/2);

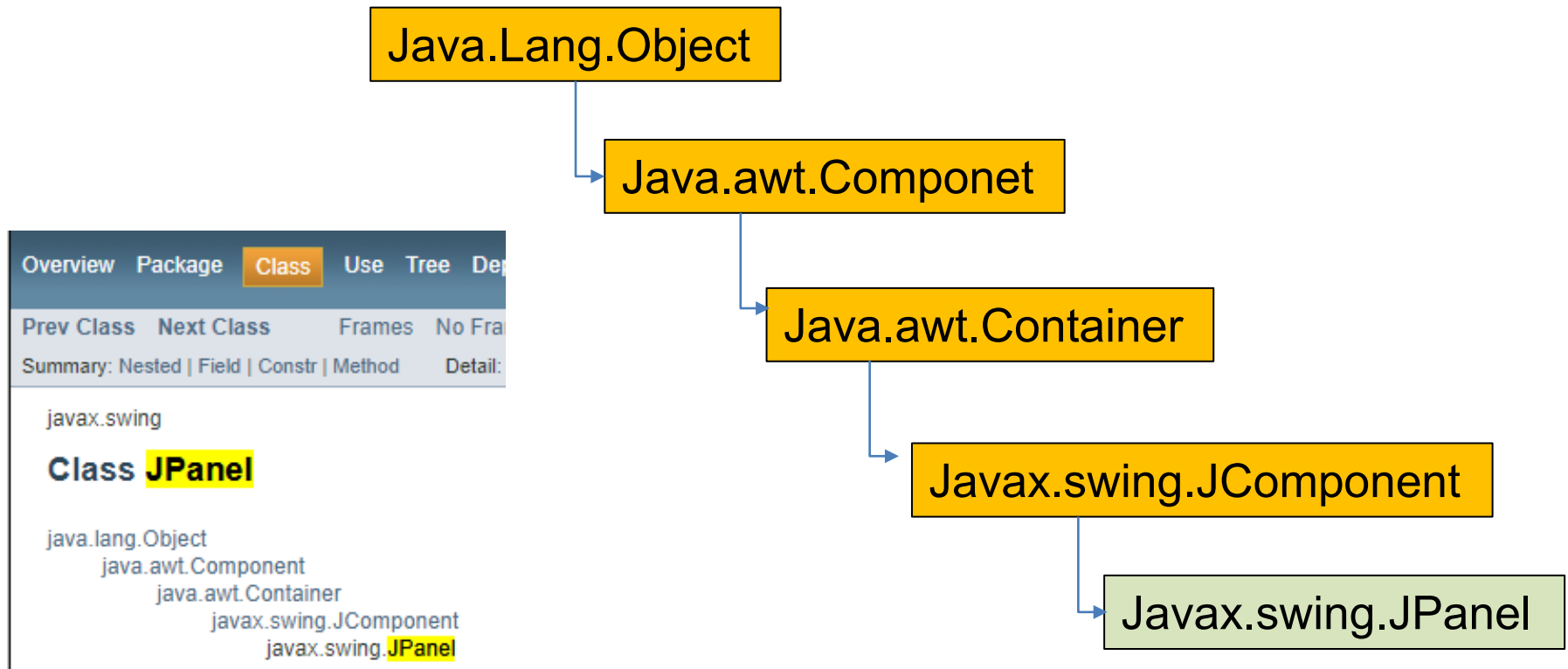
        Image icono = pantalla.getImage("src/graficos/imagen.png"); //Definir un icono en el Frame
        setIconImage(icono);

        setBounds(400,200,300,400); //Define localización y dimensión
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //acción al cerrar el Frame

        //setExtendedState(Frame.MAXIMIZED_BOTH); //maximiza la pantalla
        setTitle("TLPI-Swing"); //Título del Frame
        setResizable(false); //Permite ampliar o no el Frame
    }
}
```


Escribir en el Frame

Dentro de un Frame es posible cargar los componentes (Texto, Botones, Menús, etc) se recomienda definir un Lienzo o capa. Para esto utilizamos la clase JPanel



Escribir en el Frame

Para la definición del Lienzo, se crea una nueva clase con nombre LienzoPrincipal extendiendo la clase JPanel.

```
public class LienzoPrincipal extends JPanel
{
    public LienzoPrincipal()
    {
    }
}
```

Solo falta indicarle al Frame que tendrá un lienzo, esto se hace instanciando la clase LienzoPrincipal en la clase VentanaPrincipal

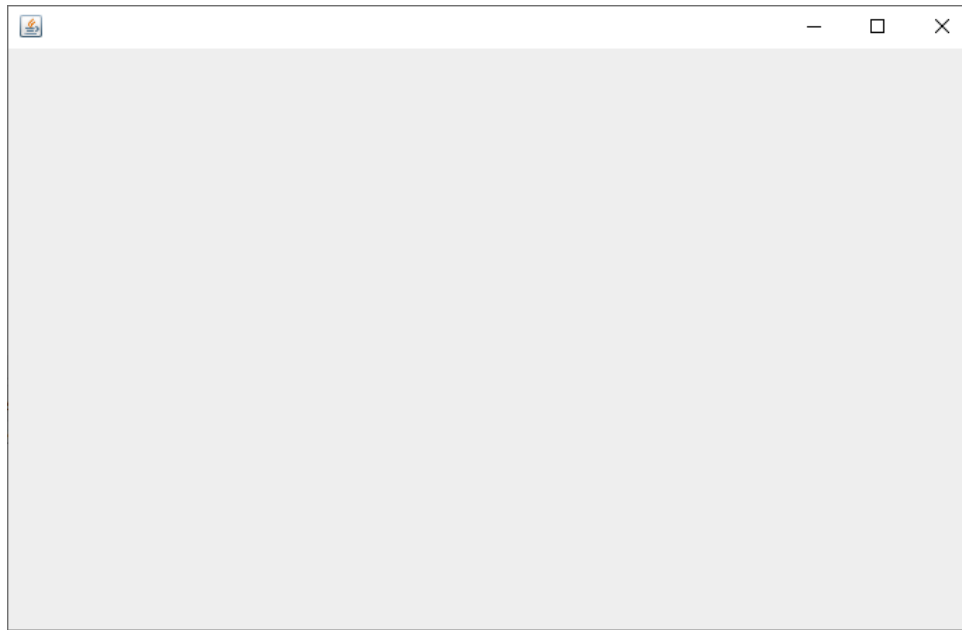
Escribir en el Frame

```
package llamadoClaseEJB;
import javax.swing.JFrame;
public class VentanaPrincipal extends JFrame{
    public VentanaPrincipal() {
        setLocation(800,400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500,300);

        LienzoPrincipal lienzoP = LienzoPrincipal();
        add(lienzoP);
        setVisible(true);
    }
}
```

Escribir en el Frame

Al ejecutar el programa el resultado es un Frame y por encima tiene un lienzo



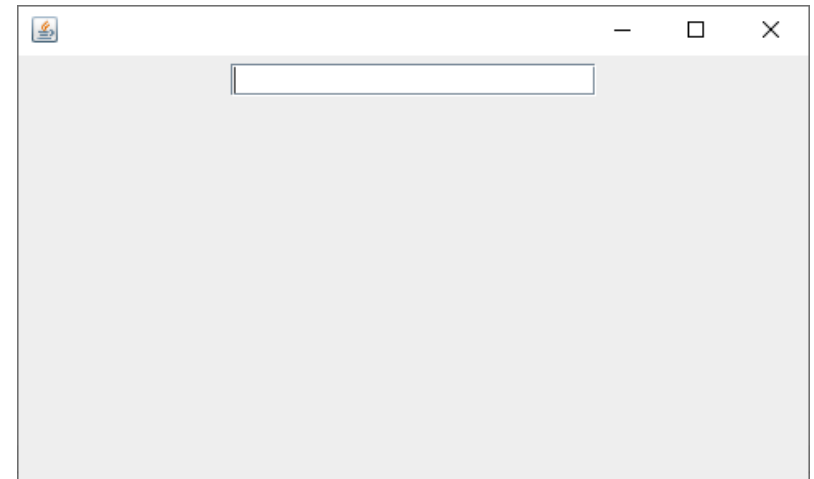
Ya es posible pintar componentes gráficos dentro del lienzo

Escribir en el Frame

Componentes

- Caja de Texto

```
public class LienzoPrincipal extends JPanel
{
    public LienzoPrincipal()
    {
        JTextField wusuario = new JTextField(20);
        add(wusuario);
    }
}
```



Escribir en el Frame

Componentes

- Botón

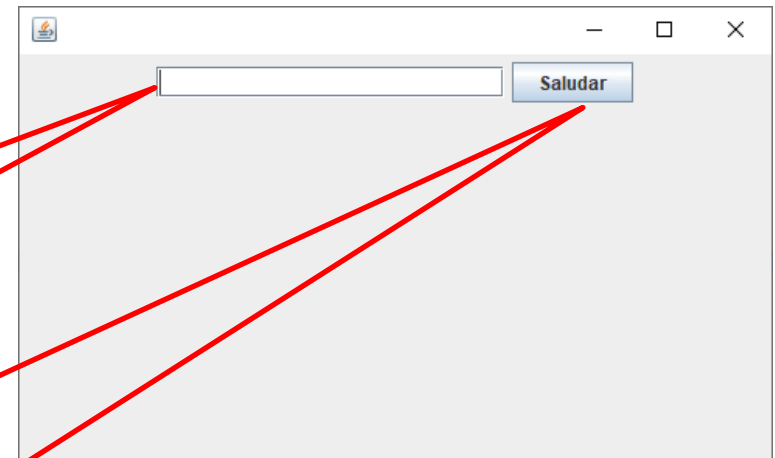
```
public class LienzoPrincipal extends JPanel
```

```
{  
    public LienzoPrincipal()  
    {
```

```
        JTextField wusuario = new JTextField(20);  
        add(wusuario);
```

```
        JButton boton1 = new JButton("Saludar");  
        add(boton1);
```

```
    }  
}
```



Escribir en el Frame

Programar un evento al componente JButton, que al dar clic llame el EJB y muestre un JOptionPane dando la Bienvenida.

Así:

Crear una clase privada llamada LlamarSaludo que implemente la Interface ActionListener, dentro de la clase LienzoPrincipal.

Escribir en el Frame

```
public class LienzoPrincipal extends JPanel {  
    public LienzoPrincipal() {  
        JTextField wusuario = new JTextField(20);  
        add(wusuario);  
  
        JButton boton1 = new JButton("Saludar");  
        add(boton1);  
  
    }  
  
    private class LlamarSaludo implements ActionListener{  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            // TODO Auto-generated method stub  
        }  
    }  
}
```

Definir un constructor que reciba el texto capturado en la caja de texto (JTextField) wusuario

Escribir en el Frame

```
public class LienzoPrincipal extends JPanel {  
    public LienzoPrincipal() {  
        JTextField wusuario = new JTextField(20);  
        add(wusuario);  
  
        JButton boton1 = new JButton("Saludar");  
        add(boton1);  
    }  
  
    private class LlamarSaludo implements ActionListener{  
  
        public LlamarSaludo(String texto) {  
            // No se programa acción en este método, solo es útil para instanciar el clase  
        }  
  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            // TODO Auto-generated method stub  
        }  
    }  
}
```

Invocar la el EJB desde el método actionPerformed

Escribir en el Frame


```
public class LienzoPrincipal extends JPanel {
    public LienzoPrincipal() {
        JTextField wusuario = new JTextField(20);
        add(wusuario);

        JButton boton1 = new JButton("Saludar");
        add(boton1);
    }

    private class LlamarSaludo implements ActionListener{

        public LlamarSaludo(String texto) {
            // No se programa acción en este método, solo es útil para instanciar el clase
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                SaludoBeanRemote saludoBean = (SaludoBeanRemote)
                    InitialContext.doLookup("/claseEJB/SaludoBean!servicio.SaludoBeanRemote");
                JOptionPane.showMessageDialog(null, saludoBean.saludo(wusuario.getText()));
            } catch (NamingException e1) {
                e1.printStackTrace();
            }
        }
    }
}
```



Se implementa un JOptionPane en el método actionPerformed que muestre el saludo, esta acción obliga a que el componente JTextField este definida a nivel de clase y no de método

Escribir en el Frame

```
public class LienzoPrincipal extends JPanel {
    JTextField wusuario;
    public LienzoPrincipal() {
        wusuario = new JTextField(20);
        add(wusuario);

        JButton boton1 = new JButton("Saludar");
        add(boton1);
    }

    private class LlamarSaludo implements ActionListener{

        public LlamarSaludo(String texto) {
            // No se programa acción en este método, solo es útil para instanciar el clase
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                SaludoBeanRemote saludoBean = (SaludoBeanRemote)
                    InitialContext.doLookup("/claseEJB/SaludoBean!servicio.SaludoBeanRemote");
                JOptionPane.showMessageDialog(null, saludoBean.saludo(wusuario.getText()));
            } catch (NamingException e1) {
                e1.printStackTrace();
            }
        }
    }
}
```

Escribir en el Frame

En la clase LienzoPrincipal, despues de la definición del Jbutton, instanciar la clase privada LlamarSaludo, enviando como parametron lo que se digite en el JTextField

```
public class LienzoPrincipal extends JPanel {  
    JTextField wusuario;  
    public LienzoPrincipal() {  
        wusuario = new JTextField(20);  
        add(wusuario);  
  
        JButton boton1 = new JButton("Saludar");  
        LlamarSaludo saludar = new LlamarSaludo(wusuario.getText());  
        boton1.addActionListener(saludar);  
        add(boton1);  
    }  
}
```