	**Escuela Superior de Empresa, Ingeniería y Tecnología.** **Ingeniería Informática**
CICLO I	**GUIA DE LABORATORIO #5** **Ingeniería de Software** **Creación de interfaces gráficas con Netbeans**

I.OBJETIVOS

- Que el estudiante pueda crear interfaces gráficas en NetBeans IDE 7.1 o superior.
- Que el estudiante aplique los MDI para llamar a jframe internos.

II. INTRODUCCIÓN

Introducción a la generación de interfaces de usuario:

El "Constructor" de interfaces de usuario del IDE NetBeans (conocido anteriormente como el "Proyecto Matisse") es un módulo del Entorno de Desarrollo Integrado NetBeans.

Este editor de interfaces gráficas está orientado hacia la librería gráfica Swing de Java. Es decir, que únicamente produce código fuente para Java.

En NetBeans 6.1 o superior el generador de interfaces gráficas de usuario se ha hecho más eficiente: ahora es más potente e intuitivo, y permite a los usuarios generar interfaces gráficas de usuario de aspecto profesional sin necesidad de profundizar en el conocimiento de los administradores de diseño.

III. PROCEDIMIENTO

Parte I: Creación de interfaz gráfica básica sin la ayuda El "Constructor" de interfaces de usuario del IDE NetBeans

1. Para empezar necesitamos crear un nuevo proyecto con el nombre **Guia5POO1.**
2. Creará la clase Main por defecto y se verá como la siguiente imagen (esto usted ya lo conoce).

```
package helloworld;

/**
 *
 * @author Magus
 */
public class Main {

    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

3. Dentro del método **main(),** escribe el siguiente código:

```java
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    System.out.println("Hola!");
}
```

Este código mostrara un mensaje que diga **Hola!** al usuario en la consola, pero esto no es lo que queremos que vea un cliente final. Para que un programa sea sencillo, es necesario que tenga una interfaz gráfica.Vamos a utilizar Swing para crear una interfaz gráfica que nos muestre el mismo mensaje.

Swing es un conjunto de librerías con las que cuenta Java para crear y mostrar una interfaz gráfica. Dentro de estas librerías hay varias clases (recuerde, una clase es como un molde con el que podemos hacer objetos) que nos permiten mostrar ventanas, mensajes, botones, cajas de texto e incluso imágenes, audio o video. Una de las clases más importantes de Swing es JFrame. Esta clase es una ventana que tiene un contenedor en el que podemos poner controles.

JFrame es una ventana normal de Windows. Dentro de un JFrame existe algo llamado ContentPane. Un **ContentPane** es un contenedor de controles en el que podemos agregar los elementos de la interfaz gráfica.

JLabel es una etiqueta con la que podemos mostrar texto en nuestras aplicaciones. Permite agregar texto, cambiar el formato, la posición, agregar imágenes y muchas cosas más.

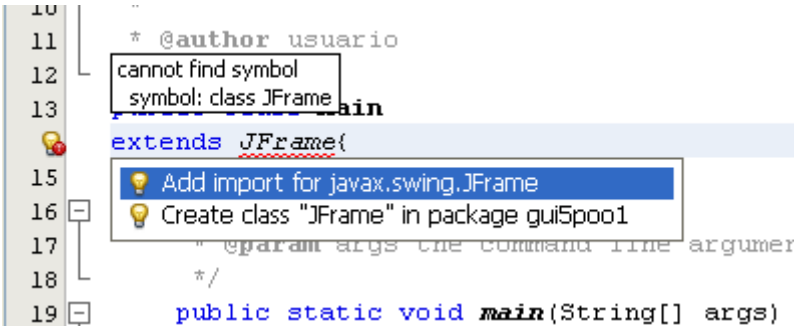
Vamos a crear una clase que sea un JFrame y después vamos a agregarle un JLabel para mostrar nuestro mensaje.

4. Vuelva al código de la clase Main. Como dijimos tenemos que convertir esta clase en un JFrame. Esto es sencillo, lo único que debemos hacer es primero, importar javax.swing.JFrame y agregar la línea extendsJFrame en la parte de arriba después de dónde dice publicclassMain como se ve en la imagen.

```
5  package gui5poo1;
6
7
8
9  /**
10  *
11  * @author usuario
12  */
13 public class Main
    extends JFrame{
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         // TODO code application logic here
21
22     }
23 }
```

Es importante notar que dejar un renglón entre el nombre de la clase y el extends no es necesario, pero ustedes pueden seguir cualquier estilo que consideren más claro. Después de escribir esta línea NetBeans la va a subrayar con una línea roja, esto significa que el código tiene un error. Poniendo el mouse sobre el error podemos obtener más información sobre él.

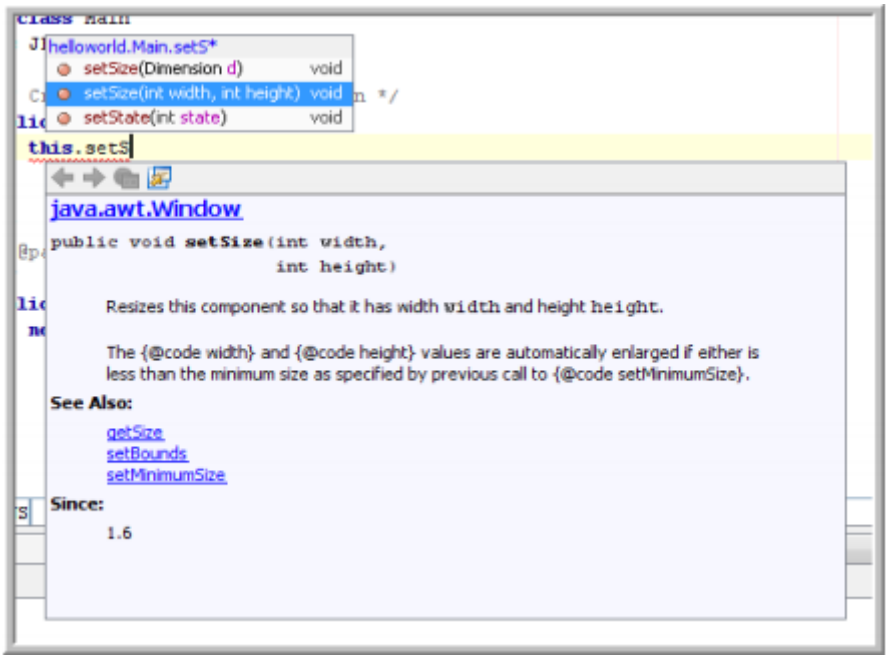
5. El error que nos marca NetBeans es que no conoce el símbolo. Esto quiere decir que no sabe qué es un JFrame.Para resolver este tipo de errores debemos agregar la librería que mencionábamos hace unos momentos.Tome en cuenta que no sabemos dónde está, pero por suerte NetBeans también nos muestra un foquito a la izquierda en el que, al hacer click, nos brinda una solución. No siempre podemos confiar en las soluciones que brinda NetBeans porque después de todo es una herramienta y no una persona que está haciendo la recomendación, pero muchas veces su sugerencia es acertada, como en este caso que dice que debe agregar un import. Seleccione la opción y debe aparecer una línea al principio de nuestra clase en la que resuelve el error.



6. Como hemos visto el método Main (con mayúscula) es el *constructor de la clase*. Este es el método con el que vamos a crear nuevos objetos de nuestra clase (que es una ventana). Dentro de este método debemos escribir algunas líneas de código para darle un tamaño a la ventana y para que se muestre. Copie el código que se muestra en la siguiente imagen:

```
/** Creates a new instance of Main */
public Main() {
    this.setSize(200,200);
    this.setTitle("JFrame");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(true);
}
```

En el código estamos utilizando la palabra clave this, que hace referencia a la misma clase que lo llama, es decir, a nuestra ventana. La primera línea va a cambiar el tamaño de nuestra ventana a 200 x 200. Escribathis.setS y espera unos momentos. Aparecerá una pantalla que te muestra opciones de auto completar. Esta pantalla es muy útil al programar ya que cuando no conocemos un método nos brinda información sobre qué significan los parámetros y cómo utilizarlos.



Por ejemplo, en esta imagen nos indica que el método setSize() debe recibir dos números enteros (int quiere decir número entero), y nos dice que el método va a cambiar el tamaño del componente (en este caso nuestra ventana) para que tenga el ancho y alto especificados. También nos indica que podemos conseguir más información viendo los métodos getSize(), setBounds() y setMinimumSize(). Utilizando esta herramienta complete el código para mostrar la ventana.

La segunda línea nos permite cambiar el título de nuestra aplicación. Podemos utilizar cualquier título que nos guste, lo único importante es pasarlo entre comillas porque es texto. La tercera línea le dice a nuestra aplicación qué debe hacer cuando el programa termine. En este caso nuestro constructor es únicamente una ventana, por lo que le indicamos que al cerrar la ventana la aplicación termine con EXIT_ON_CLOSE. La cuarta línea le indica a la ventana que es visible, por lo tanto se dibuja en pantalla.

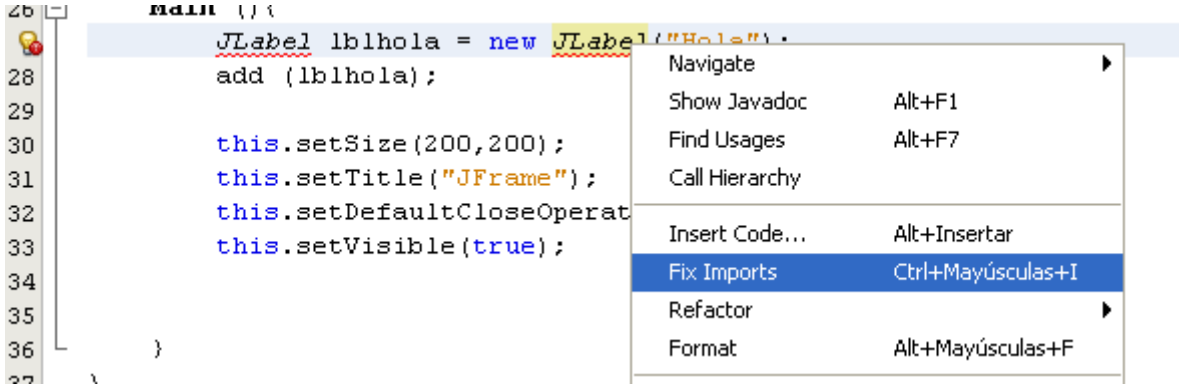
7. Por último debemos crear un nuevo objeto de nuestra ventana, esto es muy sencillo y lo logramos agregando la línea que se ve en la imagen dentro del método main() (con minúscula).

```
public static void main(String[] args) {
    new Main();
}
```

8. Con esto se muestra una ventana vacía, ya casi hacemos el programa que queríamos, pero nos falta un mensaje. Para eso vamos a utilizar otra clase de Swing que se llama JLabel. En la parte superior del constructor escriba el código como se ve en la imagen:

```java
/** Creates a new instance of Main */
public Main() {
    JLabel lblHola = new JLabel("Hola");
    add(lblHola);

    this.setSize(200,200);
    this.setTitle("JFrame");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(true);
}
```

Una vez más la primera línea va a ser subrayada en rojo porque nos falta importar una librería. Ahora vamos a utilizar otra técnica para resolver este mismo error. Haz click derecho sobre el código y selecciona la opción "FixImports". Con esto NetBeans revisará todo el código y automáticamente va a agregar todas las librearías que necesitamos.



En el código lo que estamos haciendo es crear un nuevo JLabel. Su nombre será lblHola (por convención todos los JLabel deben empezar su nombre con lbl) y este JLabel va a ser un nuevo JLabelcon el texto "Hola". Después agregamos el JLabel al contenedor de la ventana.

Resultado

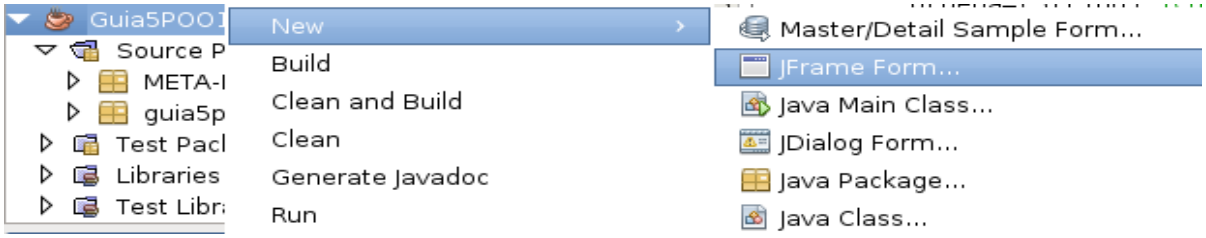
Parte II: Creación de interfaz gráfica utilizando el "Constructor" de interfaces de usuario del IDE NetBeans

Ejemplo 1

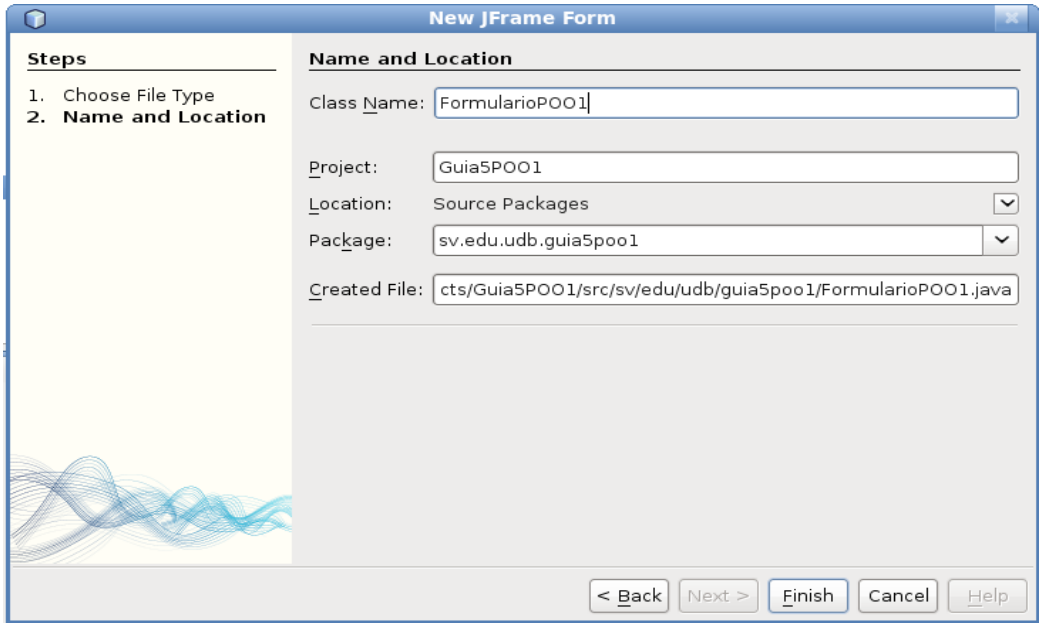
Para comenzar a generar la interfaz, debe crear un contenedor Java en el que colocar los otros componentes necesarios de la interfaz gráfica de usuario. En este paso, crearemos un contenedor utilizando el componente JFrame y lo colocaremos en un nuevo paquete.

Para crear un contenedor de Jframe:

1. En la ventana Proyectos (Project), haga clic con el botón derecho en el nodo Guia5POO1 y elija Nuevo (New) > Formulario JFrame (JFrameForm...).



- ClassName: FormularioPOO1
- Package: sv.edu.udb.guia5poo1



- Click en Finish

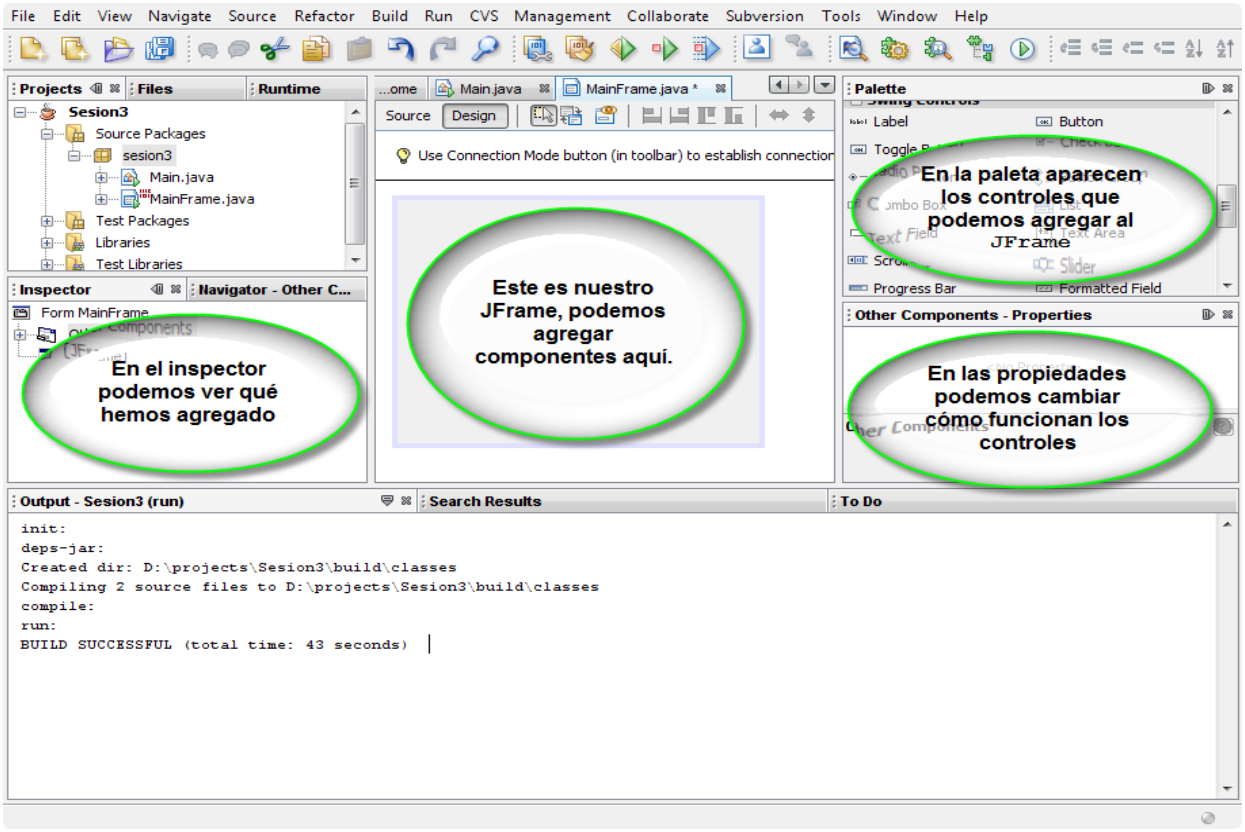
A pesar de que los nombres no son importantes para Java, se recomienda poner un nombre que exprese para qué funciona este objeto y de qué clase es, por convención debemos elegir nombres representativos que nos indiquen para qué sirve cada uno de los componentes, las primeras tres letras indiquen el tipo de componente de la siguiente manera:

Componente	Prefijo
JLabel	lbl
JButton	btn
JTextField	txt
JTextArea	txt
JPanel	pnl
JMenu	mnu
JMenuItem	mnuItem
JRadioButton	rbt

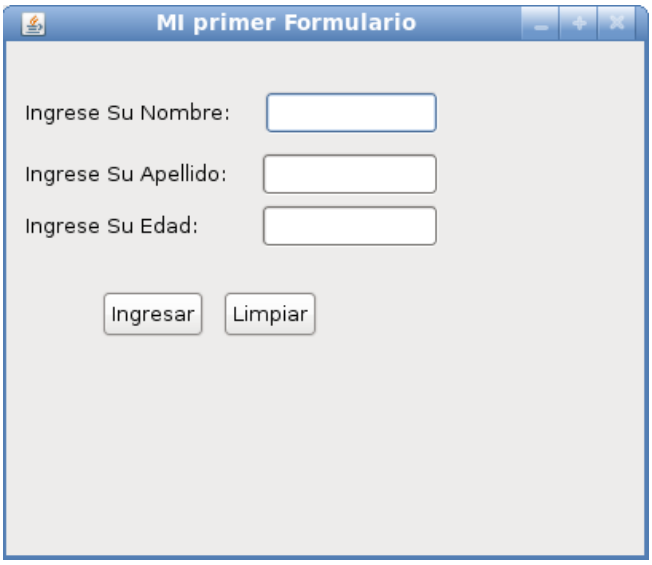
De esta manera vamos a seleccionar nombres para nuestros controles. Para cambiar los nombres debes utilizar la ventana Inspector (en versiones recientes de NetBeans la ventana Inspector se encuentra bajo el nombre de Navigator). Puede seleccionar un control y presionar F2 para cambiarle el nombre. Un ejemplo podría ser los siguientes:

Componente	Nombre
JLabel	lblName
JTextField	txtName
JButton	btnEnter

La ventana resultante esta dividida en diferentes secciones como se muestra en la figura.



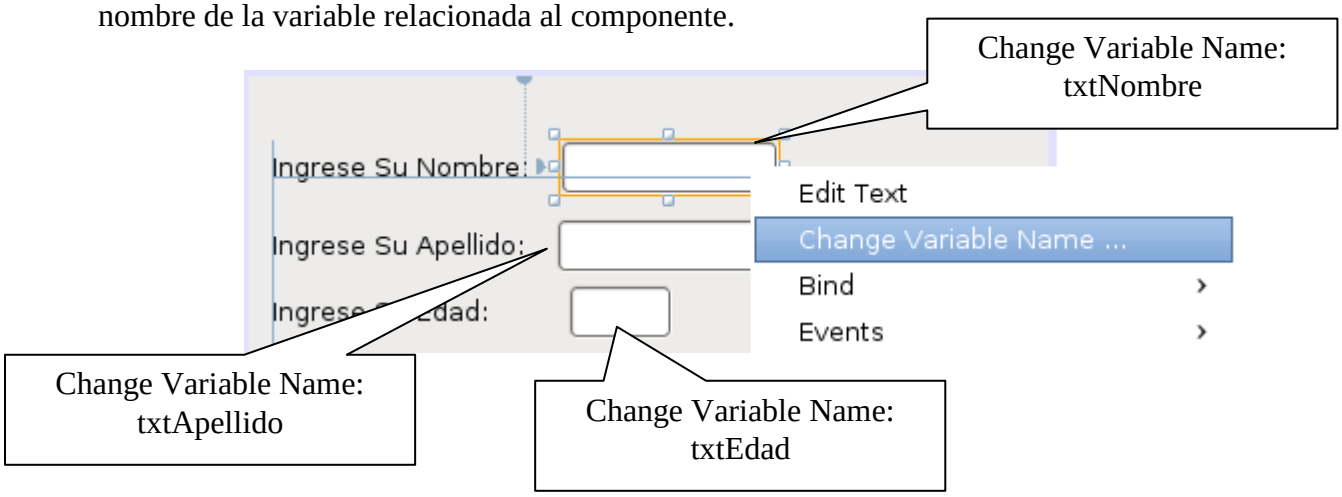
2. Ingresar al JFrame los siguientes elementos como se muestra en la siguiente figura.



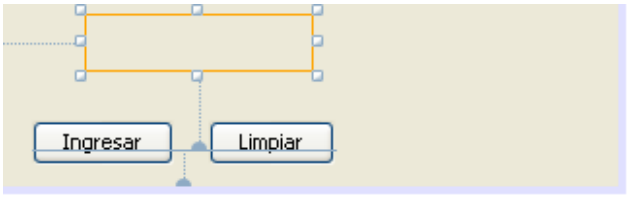
Cambiar las propiedades de cada componte según como se muestra en la siguiente tabla.

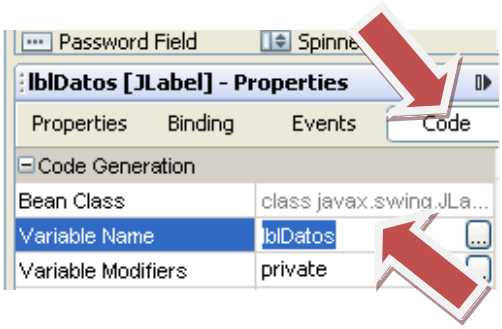
Control	Propiedad	Valor
JFrame	title	Ingreso Persona
JLabel1	text	Ingrese su nombre:
	name	lblNombre
JLabel2	text	Ingrese su apellido:
	name	lblApellido
JLabel3	text	Ingrese su edad:
	name	lblEdad
Jlabel4	text	
	name	lblDatos
JButton1	text	Ingresar
	name	btnIngresar
JButton2	text	Limpiar
	name	btnLimpiar

Nota: la propiedad name solo cambia el nombre del componente a nivel visual pero para que este también cambie a nivel de código deberá dar click derecho sobre el componente seleccionar **"Change Variable Name"**, esto permitirá abrir una nueva ventana en la cual podrá cambiar el nombre de la variable relacionada al componente.



También puede cambiar el nombre en la ficha Code de las propiedades. (Pruébelo reasignando JLabel4 por lblDatos)





3. Ya que se han agregado los controles y modificado las propiedades pasaremos al siguiente punto que consiste en agregar la funcionalidad de los botones y lo realizaremos por medio de un evento que al dar click al botón Ingresar me agregue los datos en el label**lblDatos**y a la vez me diga que si soy mayor de edad o no**,** para realizar esto debemos dar click derecho sobre el button Ingresar, seleccionar el submenú events y seleccionar el evento Action ->actionPerformed. Al hacer esto aparece una ventana con código en la que podemos agregar lo que debe realizar nuestro proyecto al hacer click en el botón.

En esta imagen nos muestra como agregar un evento.

Digite el siguiente código dentro del método **btnIngresarActionPerformed**

```
private void btnIngresarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int edad=Integer.valueOf(txtEdad.getText());
    lblDatos.setText(txtNombre.getText() +" " + txtApellido.getText() + " " + txtEdad.getText());
    if(edad>17){
        JOptionPane.showMessageDialog(this, "Eres mayor de edad");
    }
    else{

        JOptionPane.showMessageDialog(this, "Eres menor de edad");
    }
}
```

4. Ahora nuestro siguiente paso consiste en correr la aplicación, debemos dar click derecho sobre la clase FormularioPOO1.java y seleccionar la opción "**Run File".**

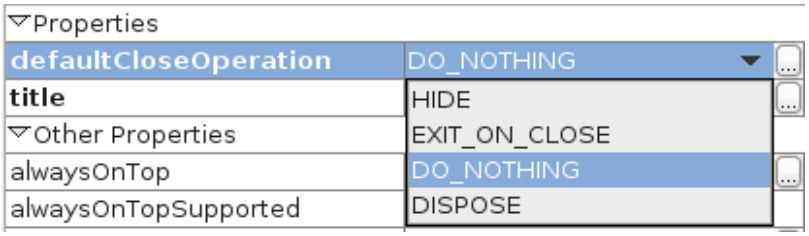
5. Al ingresar los datos y dar click en el botón **Ingresar** los valores serán pasados al label **lblDatos** y a la vez mostrara un mensaje ya que la edad es mayor que 17.

6. Si queremos que el botón Limipar tenga funcionalidad deberemos agregar el evento **ActionPerformed**de dicho botón, en el cual deberá digitar el siguiente código.

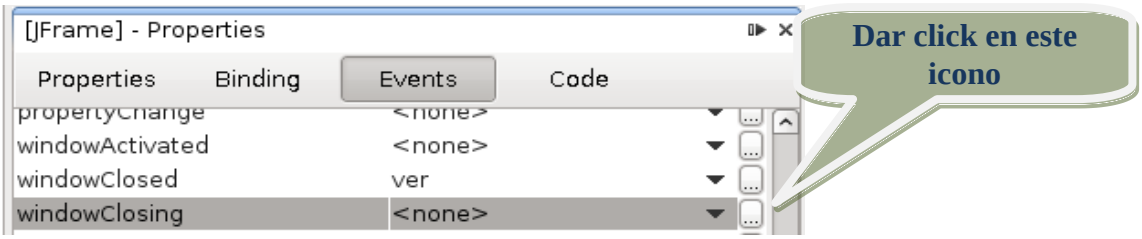
```
    private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        txtNombre.setText("");
        txtApellido.setText("");
        txtEdad.setText("");
    }
```

Nota:
Los métodos **setText()** y **getText()** nos permitirán realizar operaciones de fijar un valor para el primer método y obtener el valor con el segundo método.

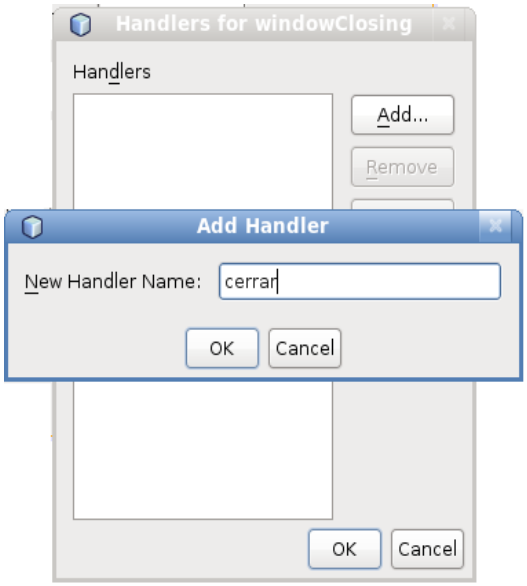
7. Ahora le daremos una mayor funcionalidad a la aplicación la cual consiste que al dar click en la "X" me pregunte si quiero salir o no para ello debe de modificar la propiedad **DefaultCloseOperation->**desplegar las opciones y seleccionar **DO_NOTHING** para eliminar la funcionalidad al dar click en la "X", si ahora intenta correr la aplicación y da click en la "X" podrá observar que no pasa nada.



8. Para poder hacer esta parte deberemos modificar el evento **windowClosing**, para ello ir a la ventana de propiedades y seleccionar la pestaña de eventos y buscar el evento **windowClosing**, tal y como se muestra en la siguiente figura.



9. Dar click en el icono mostrado en la figura, luego aparecerá una ventana como la siguiente en la cual deberemos agregar un método para que se ejecute a la hora de llamar al evento, colocar de nombre al método **"cerrar"** damos click en **OK** en ambas pantallas.



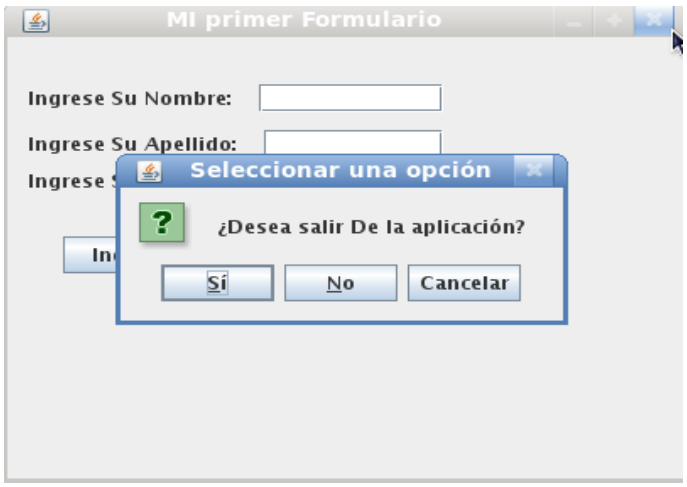
Esto agregara un nuevo método en el código como se muestra en la siguiente imagen

```
private void cerrar(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
}
```

10. Se le agregará la funcionalidad al método, agregar el siguiente código tal y como se muestra a continuación.

```
private void cerrar(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
     int result=JOptionPane.showConfirmDialog(this, "¿Desea salir De la aplicación?");

     if(result == JOptionPane.YES_OPTION) {
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
     }
}
```

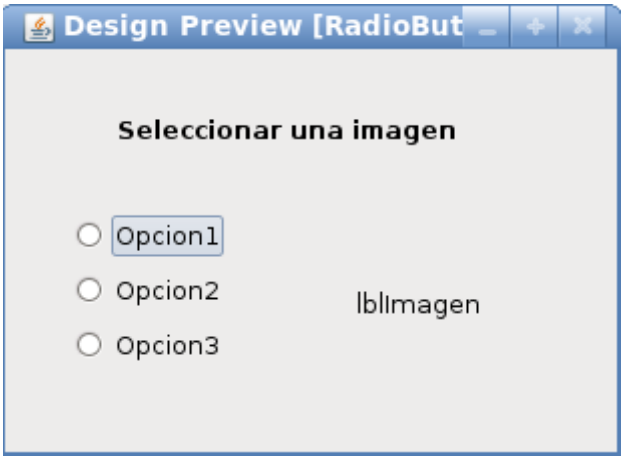
11. Si ahora intentamos correr la aplicación podremos observar que al dar click en la "X" me pregunta si Deseo salir de la aplicación.



Ejemplo2

Utilizaremos los RadioButton para poder ver diferentes imágenes.

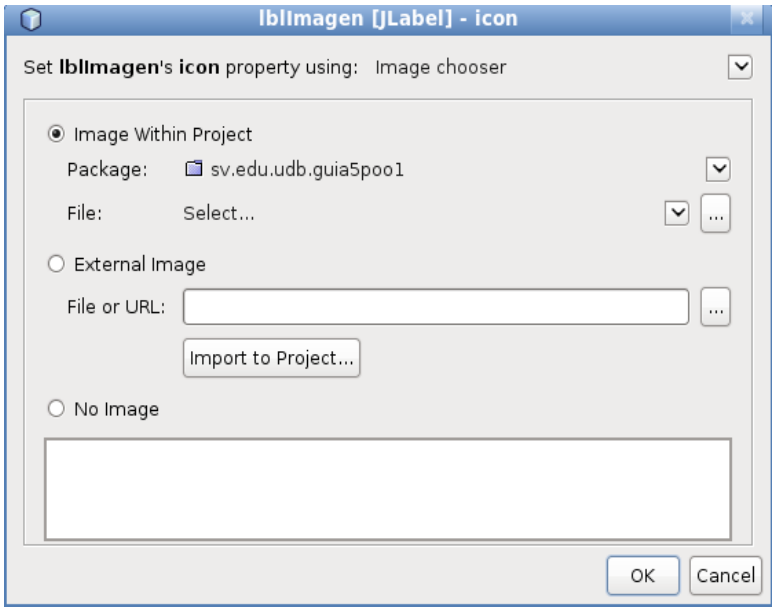
1. Agregar un nuevo Jframe con el nombre **RadioButton**.
2. Ingresar los siguientes elementos que se muestran en la imagen.



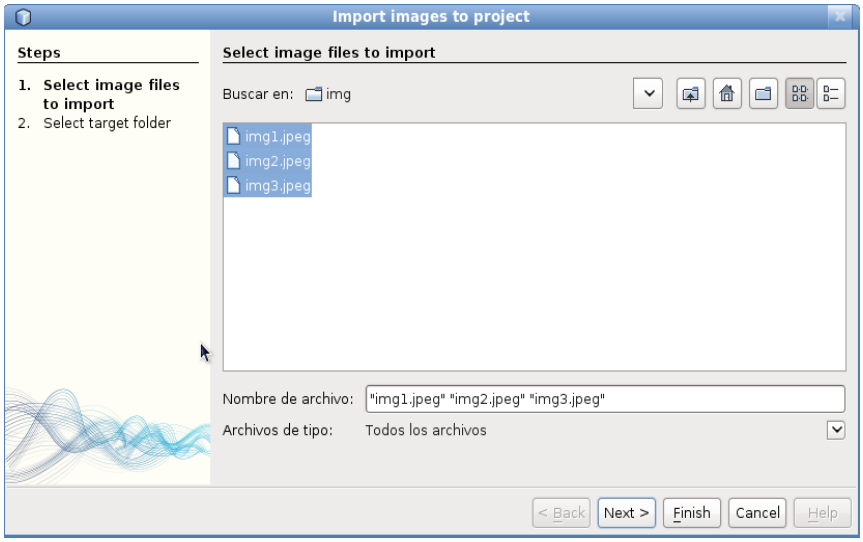
3. Modificar las propiedades según la siguiente tabla.

Control	Propiedad	Valor
JFrame	title	Uso de Radio Button
JLabel1	text	Seleccionar una imagen
	name	lblTitulo
JLabel2	text	----------
	name	lblImagen
JRadioButton1	text	Opcion1
	name	rbtOpcion1
	buttonGroup	buttonGroup1
JRadioButton2	text	Opcion2
	name	rbtOpcion2
	buttonGroup	buttonGroup1
JRadioButton3	text	Opcion3
	name	rbtOpcion3
	buttonGroup	buttonGroup1
buttonGroup1	----------	----------

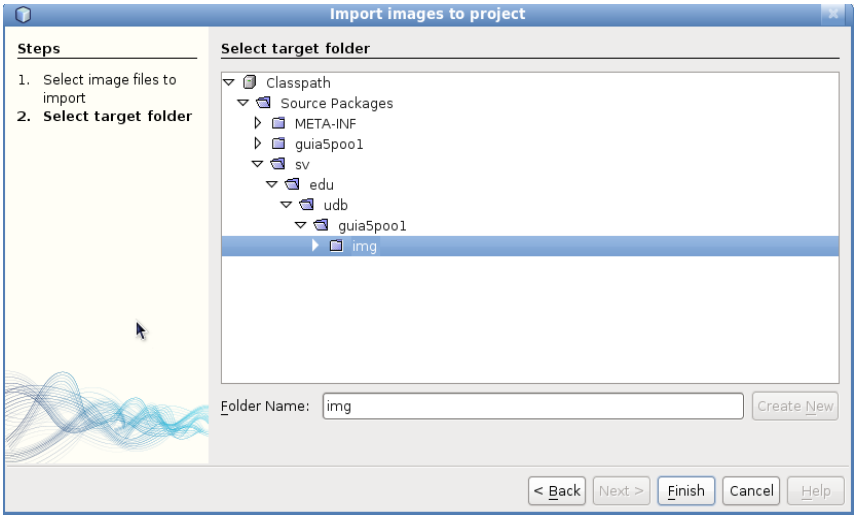
4. Para poder ver las imágenes las agregaremos al proyecto, para ello debemos dar click en la propiedad **icon**del label**lblImagen**, si damos click en el icono  y aparecerá una ventana como la siguiente.



Seleccionar **Import Project,** buscamos las imágenes y seleccionamos todas las imágenes que necesitaremos.



Finalmente seleccionamos la ubicación dentro del proyecto para este caso se creó una carpeta dentro del paquete sv.edu.udb.guia5poo1.img,en cual colocaremos las imágenes.Luego dar click en **Finish**.



5. Solo queda dar la funcionalidad a cada RadioButton, para ello dar click en el evento **ActionPerformed**de cada uno de ellos e ingresar el código tal y como se muestra en la siguiente imagen.

```
private void rbtOpcion1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
      lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource
            ("/sv/edu/udb/guia5poo1/img/img1.jpeg")));
}

private void rbtOpcion2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource
            ("/sv/edu/udb/guia5poo1/img/img2.jpeg")));
}

private void rbtOpcion3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource
            ("/sv/edu/udb/guia5poo1/img/img3.jpeg")));
}
```

Ejecútelo para ver el resultado.

Ejemplo3

Utilizaremos los **JComboBox** para poder ver diferentes imágenes.

1. Agregar un nuevo Jframe con el nombre **JComboBox**.
2. Ingresar los siguientes elementos que se muestran en la imagen.

3. Modificar las propiedades según la siguiente tabla.

Control	Propiedad	Valor
JFrame	title	Uso de JCombobox
JLabel1	text	Seleccionar una imagen
	name	lblTitulo
JLabel2	text	----------
	name	lblImagen
jComboBox1	model	Opcion1 Opcion2 Opcion3

4. Como último paso le agregaremos funcionalidad al jComboBox1 para ello seleccione el evento**item->itemStateChanged** y agregar el siguiente código.

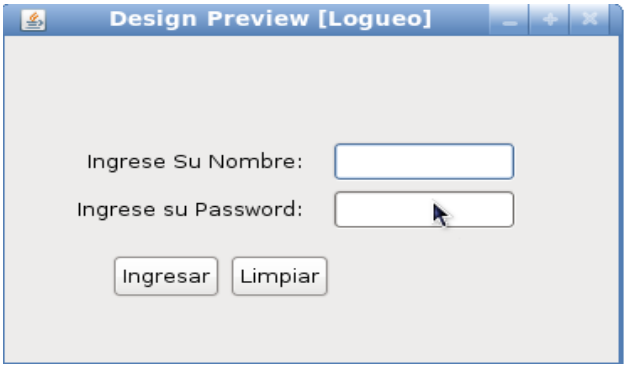
```
private void jComboBox1ItemStateChanged(java.awt.event.ItemEvent evt) {

    String prueba;
    prueba= (String)this.jComboBox1.getSelectedItem();

    if(prueba.equals("Opcion1"))
        lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource(
                "/sv/edu/udb/guia5poo1/img/img1.jpg")));

    if(prueba.equals("Opcion2"))
        lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource(
                "/sv/edu/udb/guia5poo1/img/img2.jpg")));

    if(prueba.equals("Opcion3"))
        lblImagen.setIcon(new javax.swing.ImageIcon(getClass().getResource(
                "/sv/edu/udb/guia5poo1/img/img3.jpeg")));


}
```

Ejecútelo para ver los resultados.

Ejemplo4

Para esta sección crearemos una Jframe de entrada que validaráa un usuario y un password, si estos son correctos mostrará un contenedor MDI que permitirá llamar a dos JFrame internos.

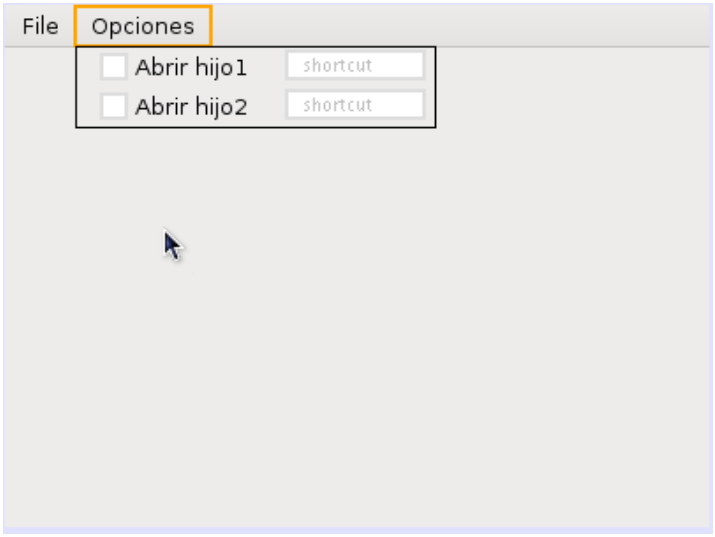
1. Para esta sección crearemos un nuevo paquete en el cual colocaremos todos los contenedores que necesitamos este paquete será llamado **"sv.edu.udb.guia5poo1.mdi"**
2. Ahora crearemos un Jframe dentro del paquete que se llame **Logueo**, agregar los componentes tal y como se muestra en la siguiente figura.



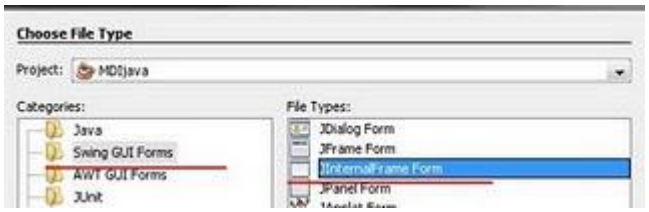
3. Cambiar las propiedades según el siguiente cuadro.

Control	Propiedad	Valor
jTextField1	name	txtNombre
jPasswordField1	name	txtPassword
jButton1	text	Limpiar
jButton2	text	Ingresar

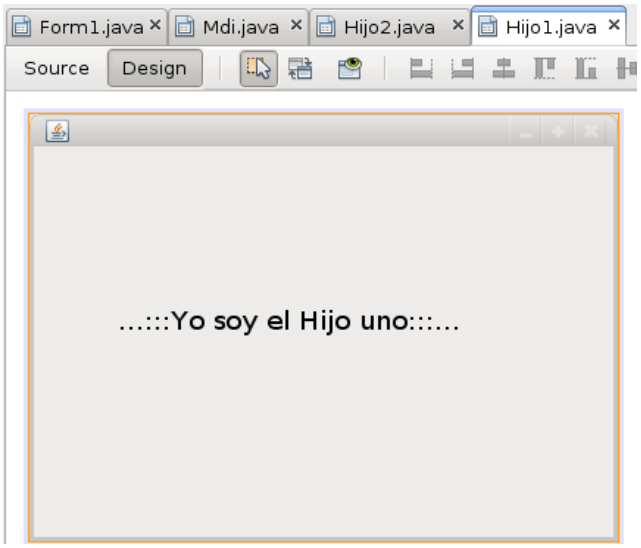
4. Ahora pasaremos agregar el MDI, damos click derecho sobre el paquete que creamos en el punto 1,seleccionamos**NEW -- OTHER --** en la ventana que nos aparece, buscamos **SWING GUI FORMS** y seleccionamos **MDI ApplicationSampleForm.** Llamarlo "**Mdi**",aparecerá un JFrameForm con controladores por defecto lo modificaremos para que nos quede similar a los siguiente.



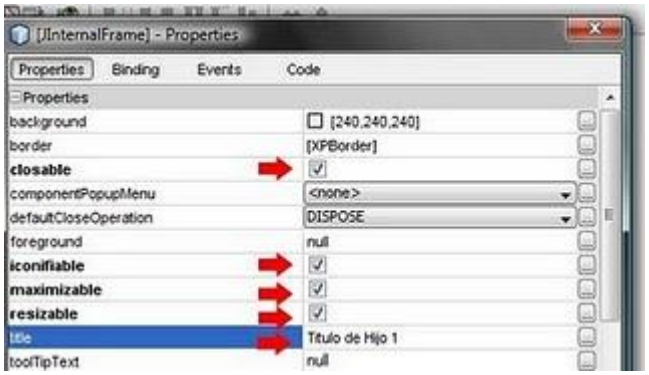
5. Como siguiente paso crearemos los JFrame internos, para ello nos dirigimos al paquete creado, hacemos click derecho y escogemos **NEW -- OTHER --** en la ventana que nos aparece, buscamos **SWING GUI FORMS y JINTERNALFRAME**, presionar siguiente y colocar un nombre al form, **"Hijo1"** y para terminar **FINISH**.



Realice este paso una vez más, pero al nuevo framellamalo**"Hijo2"**.
Esto te creará un nuevo Frame vacío, debería tener algo como esto hasta ahora



6. Dar clic derecho sobre el JInternalFrame**"Hijo1"**, elegir "**Propiedades**" (Properties) y realice los siguientes cambios :



7. Realizar los mismo pasos para el JInternalFrame "**Hijo2**"

   Puede trabajar con los nuevos frames "**Hijo1**" e "**Hijo2**" independientemente, y añadirles los objetos para interactuar en ellos.

8. Ahora a cada hijo le agregaremos un label que diga "...:::Yo soy el hijo 1:::..." y al segundo "...:::Yo soy el hijo 2:::...".

9. En este punto ya tenemos los dos hijos ahora solo debemos llamarlos para ello debe de agregar el siguiente código en el evento **ActionPerformed**en cada jMenuItem como se muestra en la siguiente figura.

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
        Hijo1 h1=new Hijo1();
        desktopPane.add(h1);
        h1.show();
}

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
        Hijo2 h2=new Hijo2();
        desktopPane.add(h2);
        h2.show();
}
```

10. Dar funcionalidad al Jframe de **Logueo**, se deberá agregar el evento **ActionPermormed**al botón de ingresar y agregar el siguiente código.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    if(txtNombre.getText().equals("rafael") && txtPassword.getText().equals("poo1")){
        new Mdi().setVisible(true);
        this.dispose();
    }
    else{
            JOptionPane.showMessageDialog(this, "Usuario ó Password incorrecto");
    }

}
```

11. Como último punto corremos el Jframe de él, ver y analizar el resultado obtenido.

Ejemplo 5

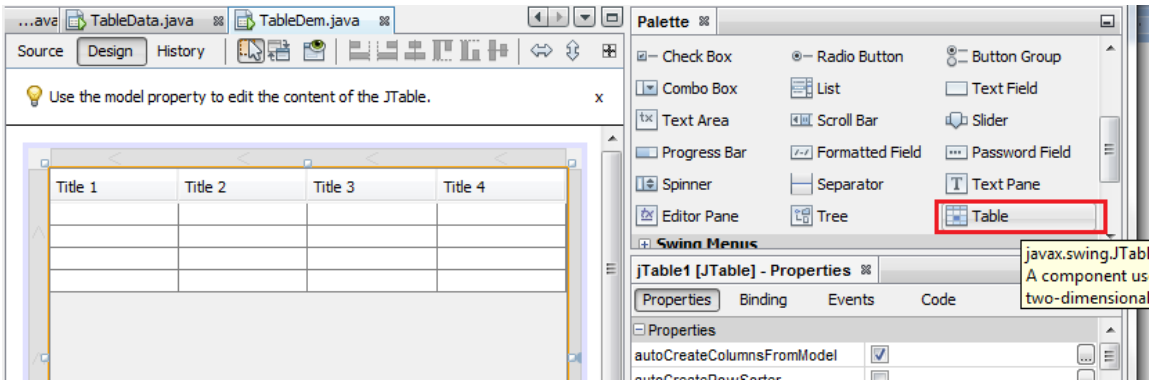
Para este punto veremos cómo utilizar un objeto que almacena un conjunto de información, como lo es el JTable, el cual es una estructura de tablas con filas y columnas.

1. Crear un nuevo JFrameForm y poner el nombre de "JTableDem".

Class Name: JTableDem

Package: sv.edu.udb.guia5poo1

2. Agregar al formulario un objeto de tipo JTable, esta se visualizará como aparece a continuación.



3. Luego en el método constructor, deberá de agregar el siguiente código, no olvidar agregar el "**DefaultTableModel**" remarcado en el recuadro rojo, las llaves en amarillo indican el inicio y fin del constructor.

```
public class JTableDem extends javax.swing.JFrame {
    DefaultTableModel dtm;
    public JTableDem() {
        initComponents();
        Object[][] datos = {
                {String.valueOf("Manuel"), String.valueOf("Hernandez"), String.valueOf("Basketball"), Integer.parseInt("5"), new Boolean(true)},
                {String.valueOf("Rafael"), String.valueOf("Torres"), String.valueOf("Futbol"), Integer.parseInt("11"), new Boolean(true)}
        };

        String[] columnNames = {"Nombre", "Apellido", "Pasatiempo", "Años de practica","Soltero(a)"};

        dtm=new DefaultTableModel(datos,columnNames);
        jTable1.setModel(dtm);
    }
```

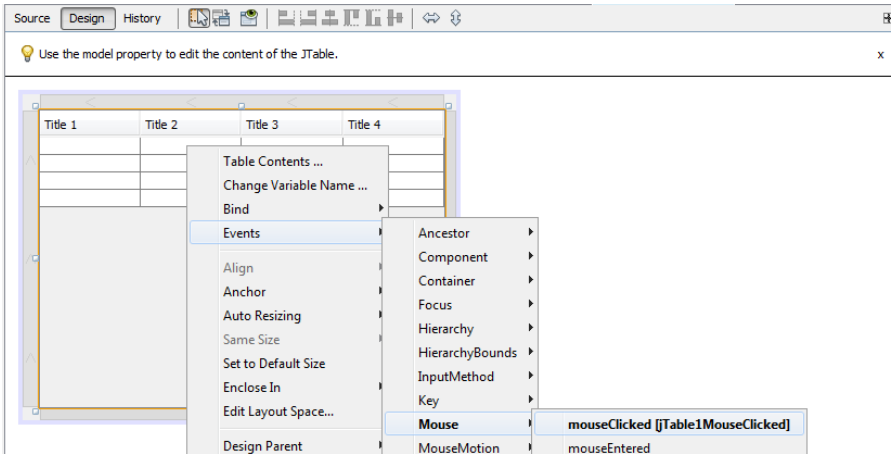
Los modelos de tabla son objetos que implementan la interface TableModel; a través de ellos es posible personalizar mucho más y mejor el comportamiento de los componentes JTable, permitiendo utilizar al máximo sus potencialidades. Todas las tablas cuentan con un modelo de tabla, existe uno por omisión.

El siguiente gráfico intenta mostrar cómo cada componente JTable obtiene siempre sus datos desde un modelo de tabla.

Relación Modelo -> Vista

La clase **AbstractTableModel** es la que implementa directamente a la interface **TableModel**, aunque es esta clase la que se recomienda extender para utilizarla como modelo de tabla, existe un modelo de tabla predeterminado que facilita mucho el trabajo con tablas. Este modelo predeterminado es la clase DefaultTableModel.

4. Como siguiente punto crear el evento "**MouseClicked**", que nos permitirá mostrar el contenido de la celda seleccionada.



5. Agregar el siguiente código al evento.

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
        int fila = jTable1.rowAtPoint(evt.getPoint());
        int columna = jTable1.columnAtPoint(evt.getPoint());
        if ((fila > -1) && (columna > -1)){
            JOptionPane.showMessageDialog(this,"Valor seleccionado: \""+dtm.getValueAt(fila,columna)+"\"");
        }
}
```

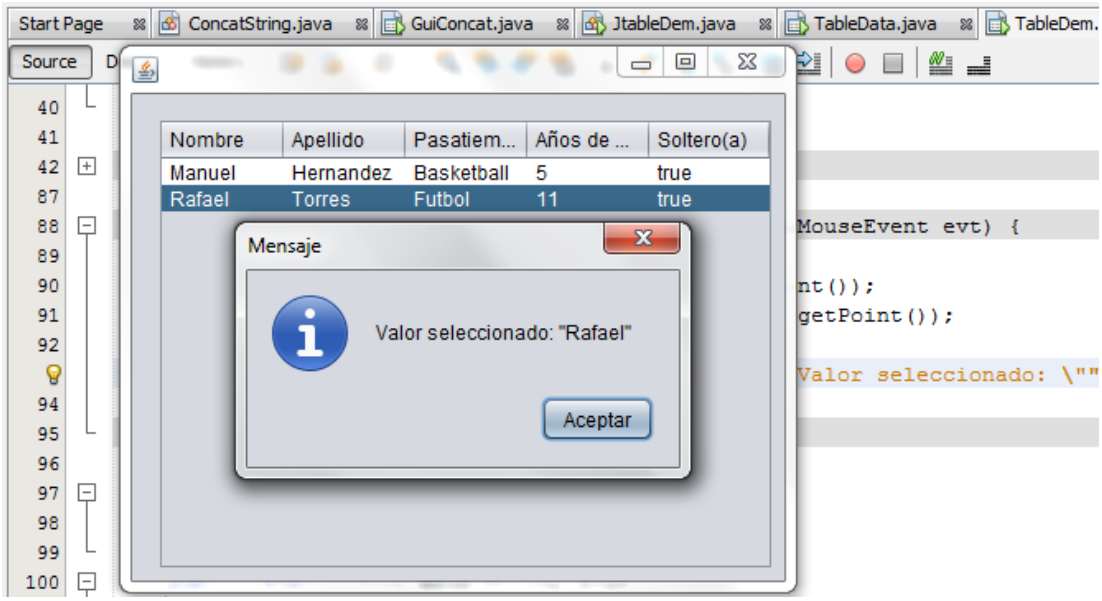
Explicación del código.
Con el método **jTable1.rowAtPoint()** es posible obtener en qué fila de del **JTable** ha ocurrido el evento del ratón (el click en este caso). Para ello basta llamar a este método pasándole las coordenadas *x*,*y*del evento de ratón, que se obtienen con el método **evt.getPoint().**

Una vez que sabemos la fila, debemos comprobar si es mayor que -1. El método **rowAtPoint()** nos devuelve -1 si pinchamos en el JTable, pero fuera de cualquier fila. Es el caso de que el JTable tenga un tamaño en pixels superior al que le corresponde según su número de filas.

Una situación similar pasa para **columnAtPoint().**

Una vez que tenemos la columna y sabemos que no es -1, es fácil a través del modelo obtener los datos correspondientes. En este caso se escribe por pantalla con un mensaje el valor de la fila y columna que se ha seleccionado.

6. Ejecutar el ejemplo y deberá ver un resultado similar a lo que se muestra a continuación



IV. EJERCICIOS COMPLEMENTARIOS

- Crear un contenedor MDI y agregar:
  - Un JINTERNALFRAME que posea radio botones para mostrar la imagen de una estrella, una galaxia o una nebulosa según la selección en un JLabel (etiqueta).
  - Un JINTERNALFRAME que permita hacer operaciones básicas (suma, resta, multiplicación y división).El tipo de operación a realizar lo debe seleccionar de un JComboBox (combobox), los operandos los debe tomar de dos JTextField (cuadro de texto) y mostrar el resultado en un JLabel (etiqueta).
  - Deberá tener un menú donde mande a llamar a todos ejemplos vistos en la clase.

- Modificar el ejemplo del JTable de tal manera que permita tener un formulario donde al seleccionar una fila de la tabla muestre los valores en el formulario para una posible edición.

V. BIBLIOGRAFIA

- http://www.javahispano.org/contenidos/archivo/63/jtable_1.pdf

Última fecha de visita: 01/03/2016

- http://docs.oracle.com/javase/tutorial/uiswing/components/table.html

Última fecha de visita: 01/03/2016

http://www.paginasprodigy.com/flabordec/00_Netbeans.html
Última fecha de visita:28/10/2016