	<p style="text-align: center;"><b>UNIVERSIDAD DON BOSCO</b> <b>ESCUELA DE COMPUTACIÓN</b></p>
<p style="text-align: center;"><b>CICLO II</b></p>	<p style="text-align: center;"><b>GUIA DE LABORATORIO #5</b> <b>Programación Orientada a Objetos</b> <b>Creación de interfaces gráficas con SWING</b></p>

## I.OBJETIVOS

- Que el estudiante pueda crear interfaces gráficas con SWING utilizando las herramientas de IntelliJ IDEA.
- Que el estudiante pueda manejar diferentes eventos de los componentes, según sea la necesidad.

## II. INTRODUCCIÓN

Java posee un juego de clases capaces de construir potentes interfaces gráficas de usuarios (GUI Graphical user Interface). A este grupo de clases se les denomina JFC Java Foundation Classes y presentan las siguientes características:

- Diseño de interfaces puramente Java y por lo tanto independientemente de la plataforma donde se ejecute.
- Extensible y personalizable: se trata de un framework intuitivo, ampliable por el programador, extendiendo e implementando las clases base. Es totalmente personalizable a nivel de look and feel, siendo posible aplicar diferentes tipos de apariencias de las ventanas, en principio, independientemente del sistema operativo sobre el que se ejecute.

La interfaz gráfica de java posee los siguientes elementos comunes:

- Contenedores: se pueden distinguir dos tipos: los contenedores base, que representan al marco de la ventana y son necesarios al menos uno por aplicación, y los contenedores que agrupan a los componentes, considerando este último tipo de contenedor como un componente que agrupa otros componentes bajo un mismo panel.
- Componente: controles que se insertarán en los contenedores (botones, cajas de texto...).
- Layouts: representan la organización de los controles sobre un contenedor.
- Gestión de eventos: Mecanismo por el cual es posible capturar y procesar los eventos del usuario producidos sobre el contenedor y sus controles.

### ¿Qué es Swing?

Es una implementación del entorno gráfico 100% basada en Java y su apariencia de visualización no es dependiendo del entorno (Sistema operativo), Swing es una biblioteca de clases que permite crear interfaces gráficas de usuario en Java.

Swing forma parte del paquete estándar, no hace falta importar ningún fichero adicional en el proyecto que sea externo de la API que ya tiene el lenguaje. No se trata de un nuevo framework de desarrollo en sistemas de ventanas, sino que más bien se basa en AWT, su aparición se dio en la versión 1.2 de Java.

La librería está implementada en 18 paquetes los más utilizados son los siguientes:

- javax.swing: es el paquete de más alto nivel, que contiene los componentes, adaptadores, los modelos por defecto de los componentes, y las interfaces para todos los modelos.
- javax.swing.event: contiene los tipos de eventos y listeners (oyentes) específicos de Swing. Además, los componentes Swing pueden generar sus propios eventos.

### 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 **InterfacesSwing**.
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:

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

¡Este código mostrará 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.

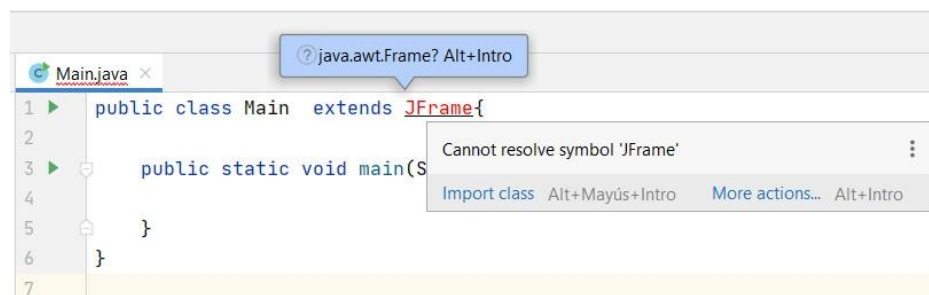


**JFrame** es una ventana normal de Windows. Dentro de un JFrame existe algo llamado **ContentPane**. Un **ContentPane** es un contenedor de controles en el que podemos agregar los elementos de la interfaz gráfica.

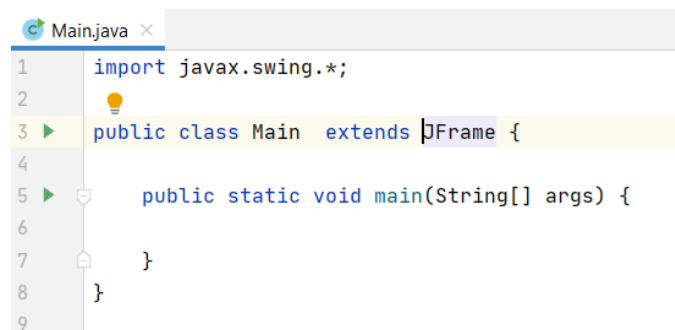
**JLabel** es una etiqueta con la que podemos mostrar texto en nuestras aplicaciones. Permite agregar texto, cambiar el formato, la posición, agregar imágenes y muchas cosas más.

Se creará una clase con un JFrame y después se le agregará un JLabel para mostrar un mensaje.

4. Regrese al código de la clase Main. Como se mencionó anteriormente se tiene que extender de un JFrame. Esto es sencillo, lo único que se debe hacer es, importar javax.swing.JFrame y agregar la línea extends JFrame en la parte de arriba después de dónde dice public class Main como se ve en la imagen.



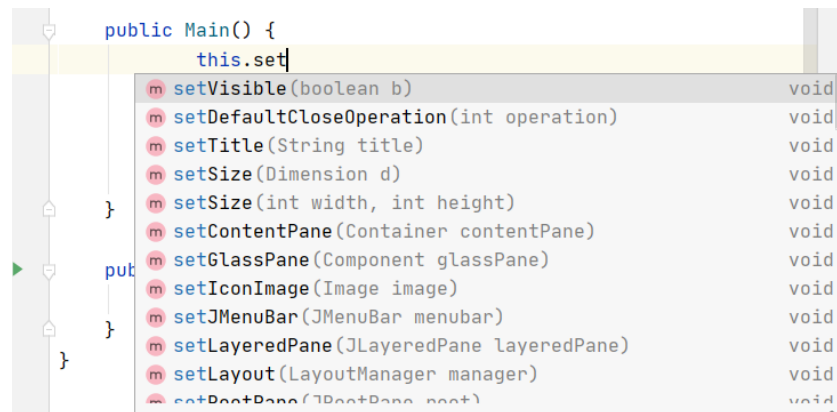
5. El error que marca es porque no se ha importado la librería de swing, para lo cual deberá agregarla manualmente o con ayuda del IDE, puede presionar “ALT+ENTER” y esto la agregará de manera automática.



6. Crear un constructor para la clase y agregar el código que se muestra en la siguiente imagen.

```
4 public class Main extends JFrame {
5
6
7     public Main() {
8         this.setSize( width: 200, height: 200);
9         this.setTitle("JFrame");
10        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        this.setVisible(true);
12    }
13 }
```

En el código se está 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 200x 200. Escriba **this.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 se conoce un método brinda información sobre qué significan los parámetros y cómo utilizarlos.



```
public Main() {
    this.set
    m setVisible(boolean b) void
    m setDefaultCloseOperation(int operation) void
    m setTitle(String title) void
    m setSize(Dimension d) void
    m setSize(int width, int height) void
    m getContentPane(Container contentPane) void
    m setGlassPane(Component glassPane) void
    m setIconImage(Image image) void
    m setJMenuBar(JMenuBar menubar) void
    m setLayeredPane(JLayeredPane layeredPane) void
    m setLayout(LayoutManager manager) void
    m setDefaultCloseOperation(int operation) void
```

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 permite cambiar el título de la aplicación. Se puede utilizar cualquier título, 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()**.

```
18 public static void main(String[] args) {
19     new Main();
20 }
```

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:

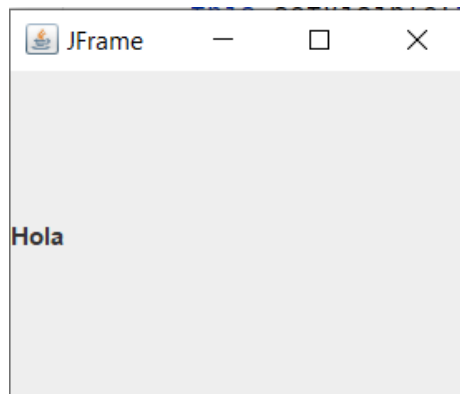
```

7      public Main() {
8          JLabel lblHolga = new JLabel( text: "Hola");
9          add(lblHolga);
10
11      this.setSize( width: 200, height: 200);
12      this.setTitle("JFrame");
13      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14      this.setVisible(true);
15
16  }

```

Lo que se termina realizando 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 JLabel con el texto "Hola". Después agregamos el JLabel al contenedor de la ventana.

## Resultado



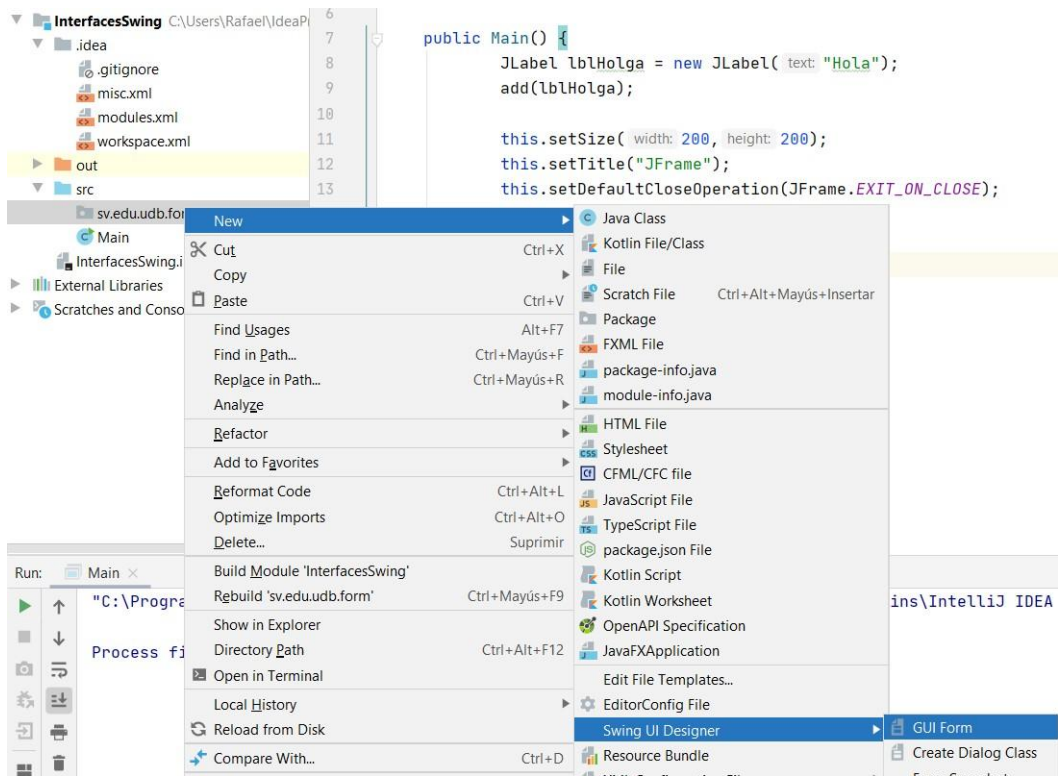
## Parte II: Creación de interfaz gráfica utilizando el “Constructor” de interfaces de usuario del IDE

### Ejemplo 1

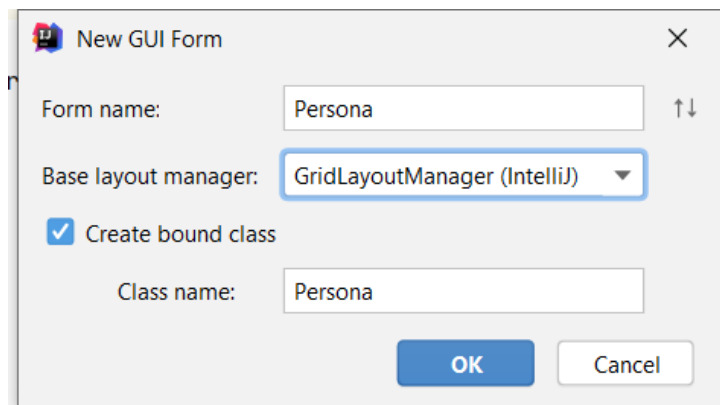
Para comenzar a generar la interfaz, debe crear un contenedor Java en el que colocar los otros componentes necesarios de la interfaz gráfica de usuario. En este paso, crearemos un contenedor utilizando el componente JFrame y lo colocaremos en un nuevo paquete.

#### Para crear un contenedor de JFrame:

1. Crear los siguientes paquetes:
  - **sv.edu.udb.form**
  - **sv.edu.udb.util**
  - **sv.edu.udb.img**
2. Sobre el paquete “sv.edu.udb.form” dar click derecho y seleccionar “NEW -> Swing UI Designer -> GUI Form”



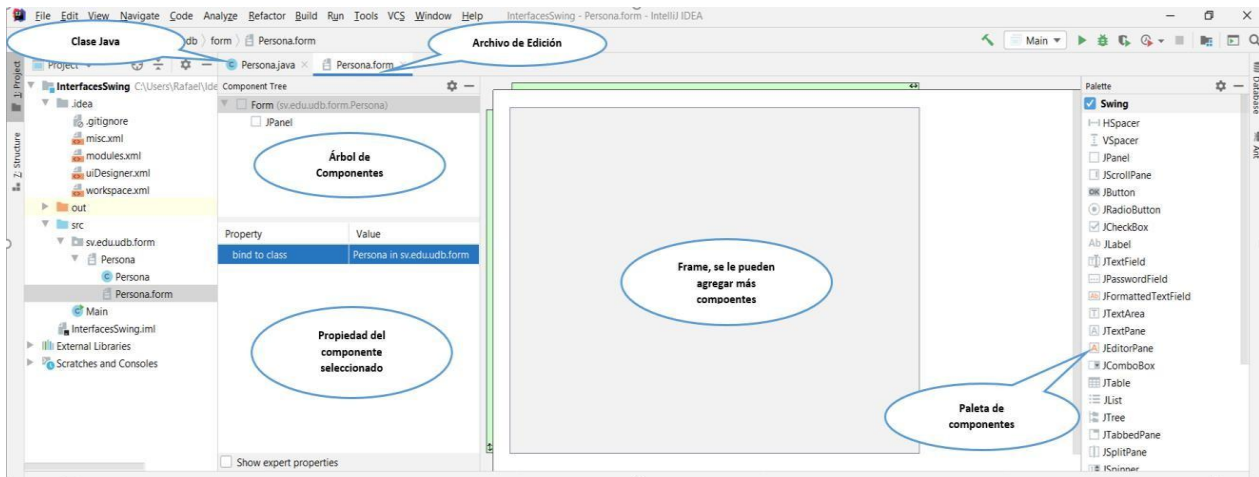
Aparecerá una ventana en la cual se debe colocar el nombre del Formulario, el nombre de la clase y la forma de manipular los componentes en el Layout, el resultado final deber ser como se muestra en la siguiente imagen



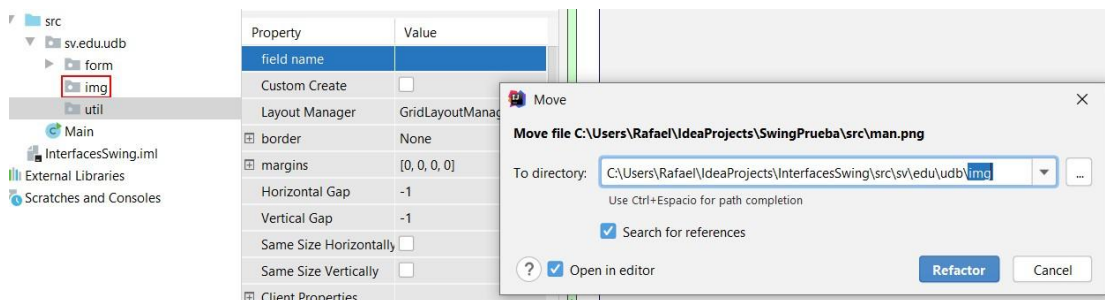
A pesar de que los nombres no son importantes para Java, se recomienda poner un nombre que exprese para qué funciona este objeto y de qué clase es, por convención debemos elegir nombres representativos que nos indiquen para qué sirve cada uno de los componentes, las primeras tres letras indiquen el tipo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

Componente	Prefijo
<b>JLabel</b>	lbl
<b>JButton</b>	btn
<b>TextField</b>	txt
<b>TextArea</b>	txt
<b>JPanel</b>	pnl
<b>JMenu</b>	mnu
<b>MenuItem</b>	mnuItem
<b>JRadioButton</b>	rbtn

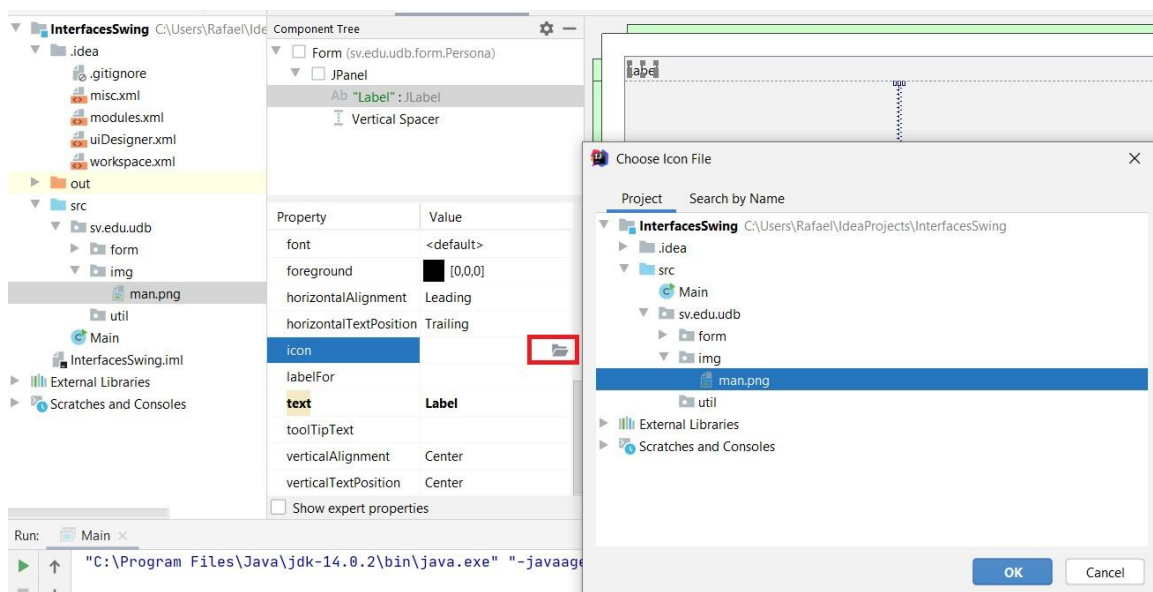
La ventana resultante está dividida en diferentes secciones como se muestra en la figura.



- Se le proporcionará una imagen llamada “man.png” la cual deberá de colocar en el paquete llamado “sv.edu.udb.img”, la forma de hacerlo es arrastrando la imagen y poniéndola dentro del paquete en el IDE, aparecerá la siguiente ventana en la cual deberá de darle Refactor y aparecerá la imagen quedará en el paquete:

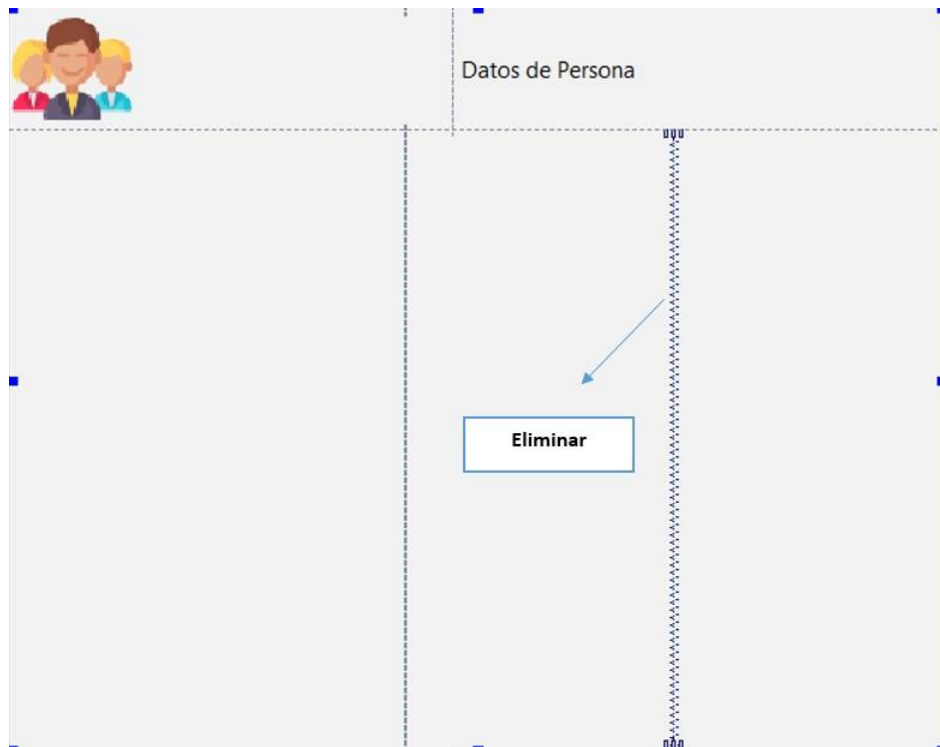


- Agregar un nuevo JLabel, buscar la propiedad text y quitarle el texto que este contenga, luego buscar la propiedad icon, seleccionar la carpeta marcada en rojo y buscar la imagen dentro del paquete. Seleccionar la imagen y dar ok.





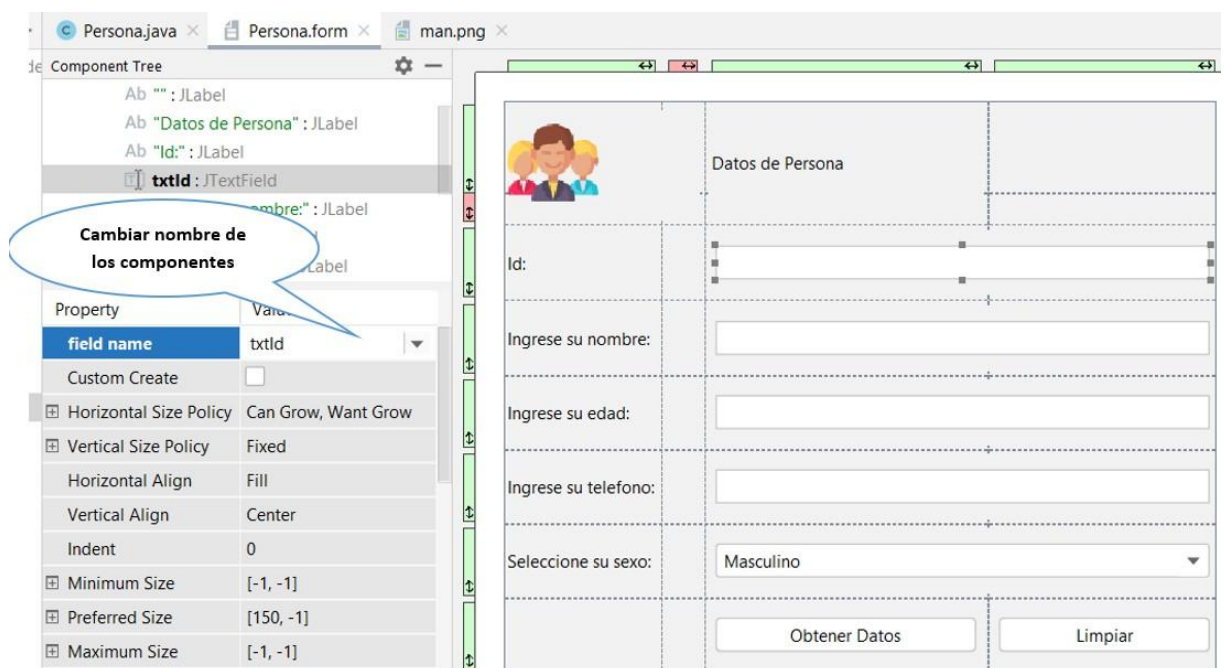
5. Agregar un nuevo JLabel y ubicarlo a la par de la imagen y agregarle el texto “Datos de Persona”, luego eliminar la rejilla de división que se muestra en la imagen.



6. Deberá de agregar los siguientes elementos:

- 5 JLabel
- 4 JTextField
- 1 JComboBox
- 2 JButton

El resultado debe ser como se muestra en la siguiente imagen.





A continuación, aparece el cuadro de las propiedades que se deben de modificar para este ejemplo.

Control	Propiedad	Valor
<b>JPanel</b>	field name	pnlPersona
<b>JLabel1</b>	text	Ingrese su id:
	field name	lblId
<b>JLabel2</b>	text	Ingrese su nombre:
	field name	lblNombre
<b>JLabel3</b>	text	Ingrese su edad:
	field name	lblEdad
<b>JLabel4</b>	text	Ingrese su teléfono:
	field name	lblTelefono
<b>JLabel5</b>	text	Selece su sexo:
	field name	lblbSexo
<b>JTextField1</b>	text	vacío
	field name	txtId
<b>JTextField2</b>	text	vacío
	field name	txtNombre
<b>JTextField3</b>	text	vacío
	field name	txtEdad
<b>JTextField4</b>	text	vacío
	field name	txtTelefono
<b>JComboBox1</b>	model	Masculino, Femenino
<b>JButton1</b>	text	Obtener Datos
	field name	btnObtenerDatos
<b>JButton2</b>	text	Limpiar
	field name	btnLimpiar

- Hasta este punto solamente se ha creado el diseño, pero no se puede correr la aplicación hasta que esta sea invocada desde la clase Persona que se creó por defecto, para ello se creara un constructor con parámetros el cual tendrá los datos de inicialización del Frame.

**Nota:** Se puede observar que la clase Persona contiene todos los elementos que se crearon en la ventana de diseño.

```
package sv.edu.udb.form;

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

public class Persona extends JFrame{
    private JTextField txtId;
    private JTextField txtNombre;
    private JTextField textField3;
    private JTextField textField4;
    private JComboBox comboBox1;
    private JButton obtenerDatosButton;
    private JButton limpiarButton;
    private JLabel lblNombre;
    private JLabel lblId;
    private JPanel pnlPersona;

    /*
```

```

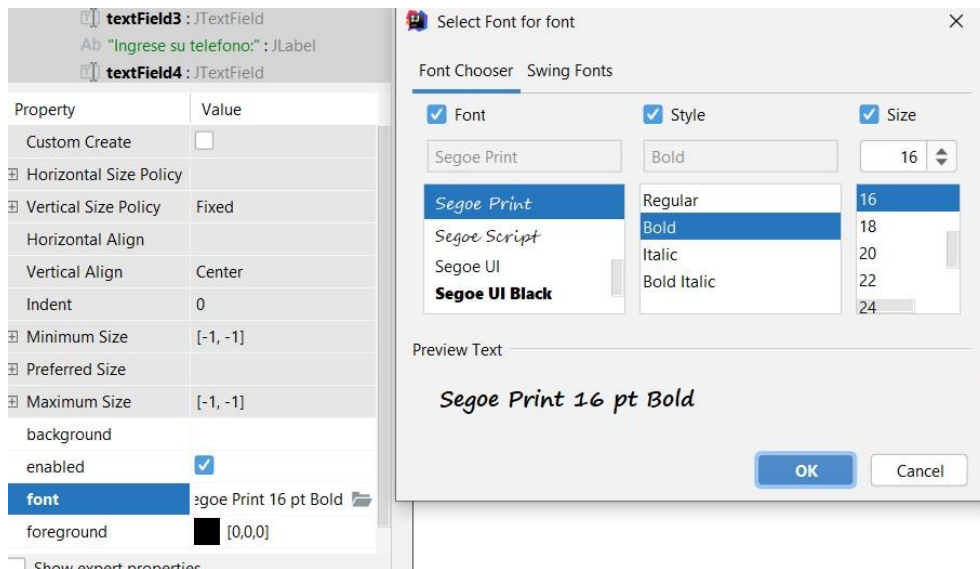
    * Constructor con Pametro para inicializar el
    * JFrame creado
    */
public Persona(String title) {
    super(title);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setContentPane(pnlPersona);
    this.setMinimumSize(new Dimension(600, 500));
    this.setLocationRelativeTo(getParent());
}

public static void main(String[] args) {
    JFrame frame = new Persona("Ingreso de Datos");
    frame.setVisible(true);
}
}

```

8. Al correr la aplicación se puede ver un resultado como el que se muestra a continuación.

9. Le cambiaremos un poco el diseño, seleccionado todos los JLabel, JTextField, JComboBox y JButton (Exceptuando el JLabel de "Datos Persona") y cambiaremos las propiedades de font, como se muestra en la siguiente imagen.

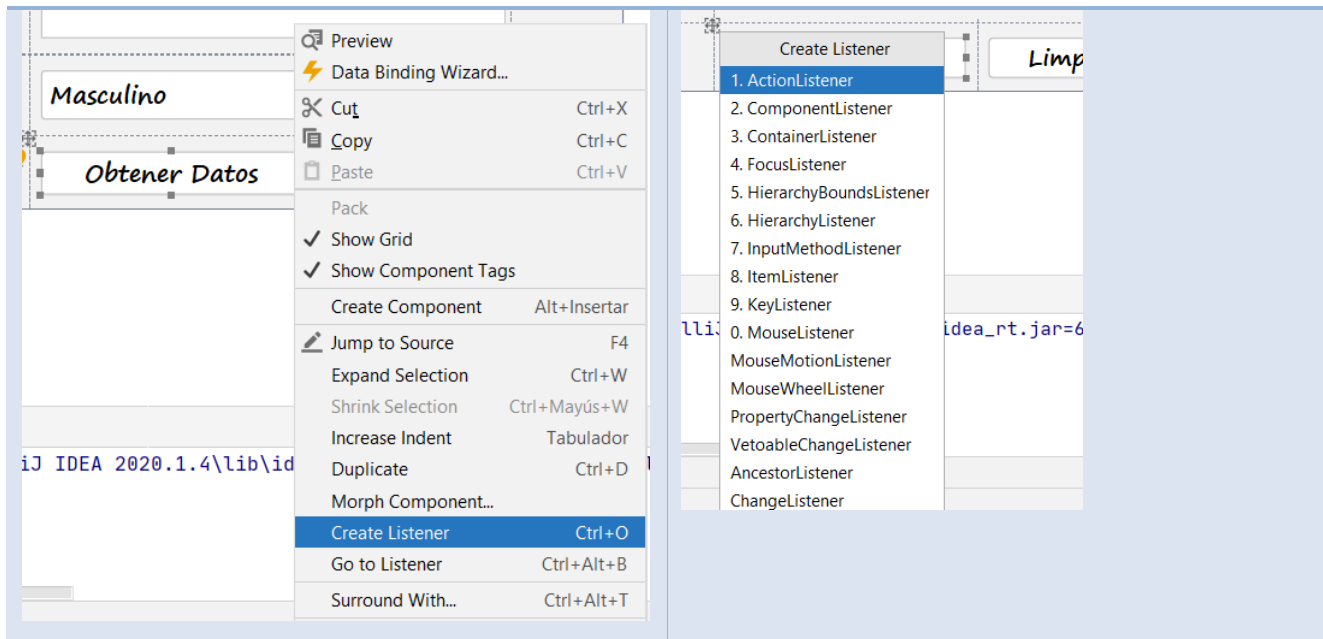


Para el JLabels “Datos de Persona”, seleccionar el mismo tipo de letra y negrita, pero el tamaño será de 36, el resultado será como el que se muestra a continuación

10. Se procederá a darle funcionalidad a los botones para ello seleccionar el botón btnObtenerDatos y darle Click derecho, luego seleccionar la opción “Create Listener”, tal y como se muestra en la siguiente imagen, aparecerá una nueva ventana con los diferentes Listener a utilizar, para este caso se deberá seleccionar “ActionListener”.

## Selección de Create Listener

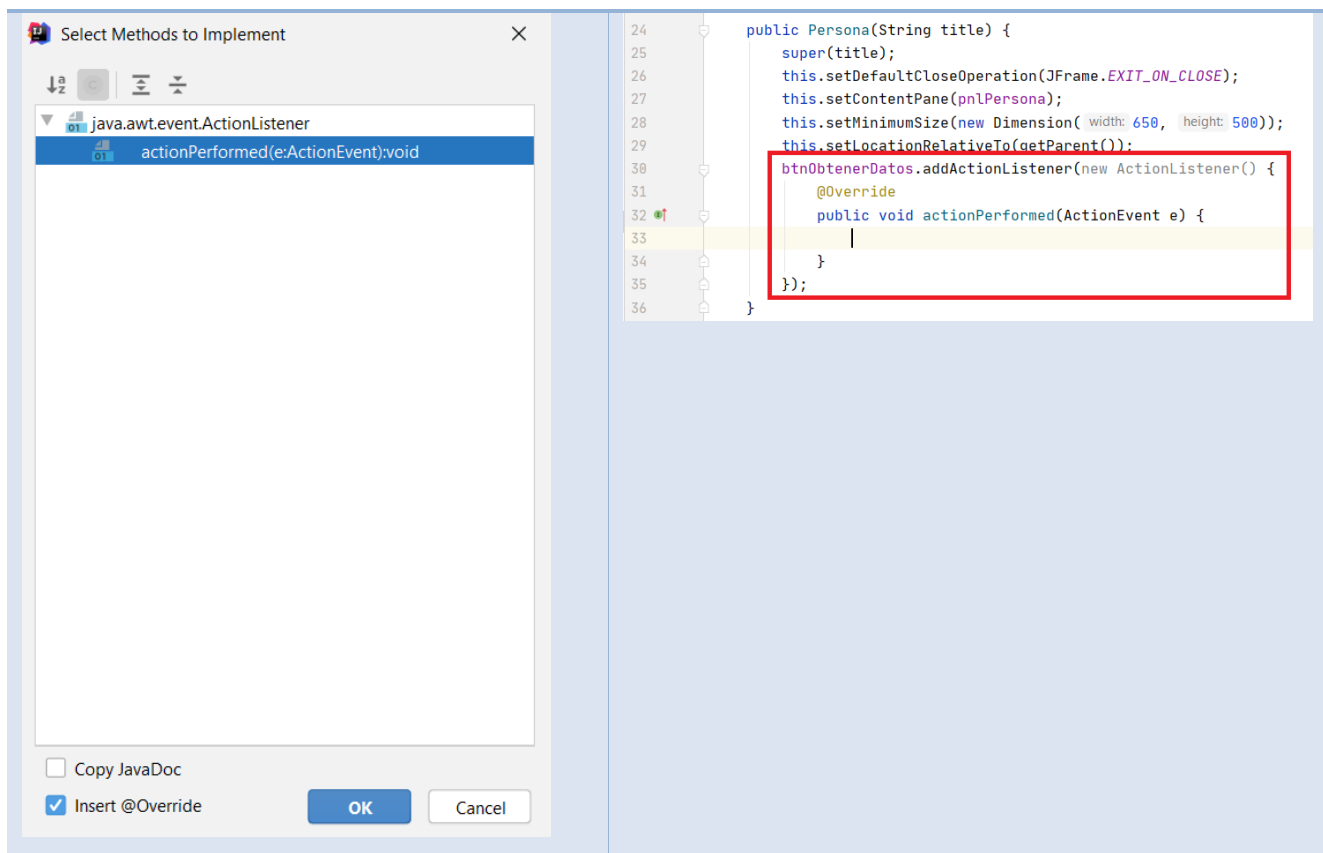
## Tipos de Listener



11. Aparecer otra ventana donde seleccionaremos el método a implementar, para este caso `actionPerformed`, dar Click en OK y esto generará un código en el constructor, el cual será invocado a la hora de dar click en el botón.

## Selección de Create Listener

## Tipos de Listener



12. En el método `actionPerformed`, crearemos la invocación a un nuevo método llamado `btnObtenerDatos`, este método será el utilizado para obtener los datos de las cajas de texto.

```
32 public void actionPerformed(ActionEvent e) {  
33     btnObtenerDatos();  
34 }  
35 }  
36 }  
37  
38 private void btnObtenerDatos() {  
39 }  
40
```

13. Agregaremos el siguiente código al método, en el cual estamos obteniendo los datos y mandándolos a pantalla con `JOptionPane`.

```
private void btnObtenerDatos() {  
    String id;  
    String nombres;  
    String edad;  
    String telefono;  
    String sexo;  
  
    id      = txtId.getText();  
    nombres = txtNombre.getText();  
    edad    = txtEdad.getText();  
    telefono = txtTelefono.getText();  
    sexo     = cmbSexo.getSelectedItem().toString();  
  
    JOptionPane.showMessageDialog(null, "Datos Obtenidos: \n ID: " + id +  
        "\n nombres: " + nombres + "\n Edad: " + edad + "\n Telefono: " + telefono +  
        "\n Sexo: " + sexo);  
}
```

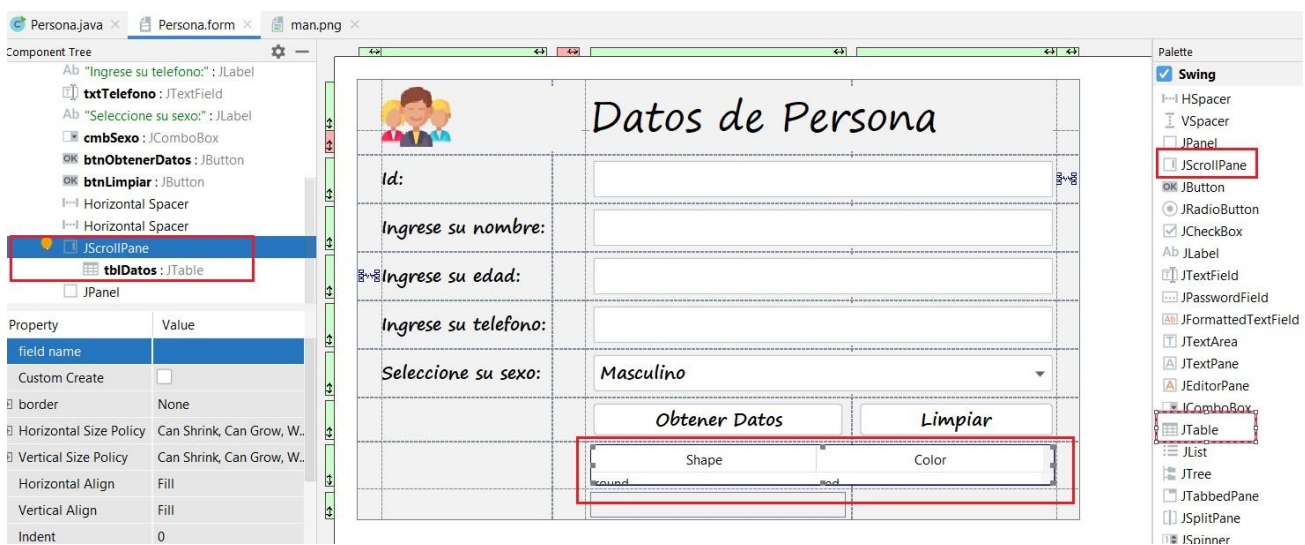
El resultado de la ejecución es el siguiente.



14. Para el botón de limpiar, crea el Listener correspondiente y crear el método llamado `btnLimpiar()` y agregar el siguiente código, el cual limpia las cajas de texto y pone el combobox en su valor de índice 0.

```
private void btnLimpiar() {  
    txtId.setText("");  
    txtNombre.setText("");  
    txtEdad.setText("");  
    txtTelefono.setText("");  
    cmbSexo.setSelectedIndex(0);  
}
```

15. Como siguiente paso, se van a agregar un **JScrollPane** y en su interior se va a agregar una **JTable** y ponerle en el field name “tblDatos”, el resultado se debe ver como se muestra en la siguiente imagen.



16. Para esta parte se utilizarán los modelos para poder dibujar la tabla. 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.

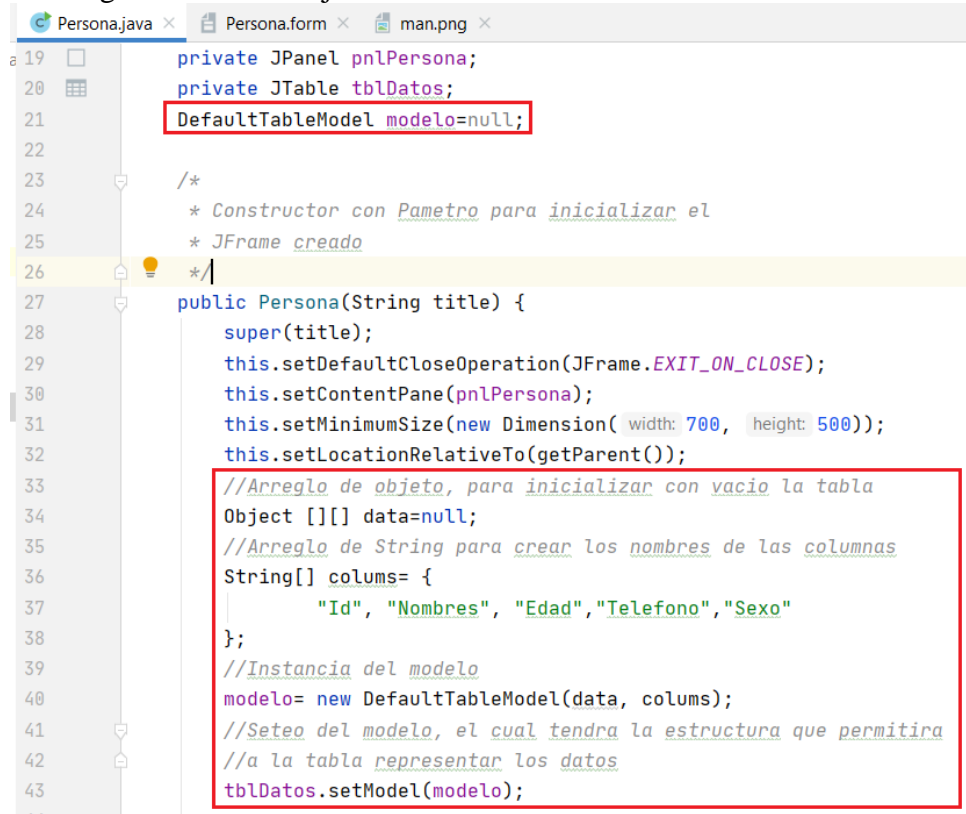
El siguiente gráfico intenta mostrar cómo cada componente `JTable` obtiene siempre sus datos desde un modelo de tabla.



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**.

Agregar el código marcado en rojo



```
19 private JPanel pnlPersona;
20 private JTable tblDatos;
21 DefaultTableModel modelo=null;
22
23 /*
24  * Constructor con Parametro para inicializar el
25  * JFrame creado
26  */
27 public Persona(String title) {
28     super(title);
29     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30     this.setContentPane(pnlPersona);
31     this.setMinimumSize(new Dimension( width: 700, height: 500));
32     this.setLocationRelativeTo(getParent());
33     //Arreglo de objeto, para inicializar con vacio la tabla
34     Object [][] data=null;
35     //Arreglo de String para crear los nombres de las columnas
36     String[] columns= {
37         "Id", "Nombres", "Edad", "Telefono", "Sexo"
38     };
39     //Instancia del modelo
40     modelo= new DefaultTableModel(data, columns);
41     //Seteo del modelo, el cual tendra la estructura que permitira
42     //a la tabla representar los datos
43     tblDatos.setModel(modelo);
```

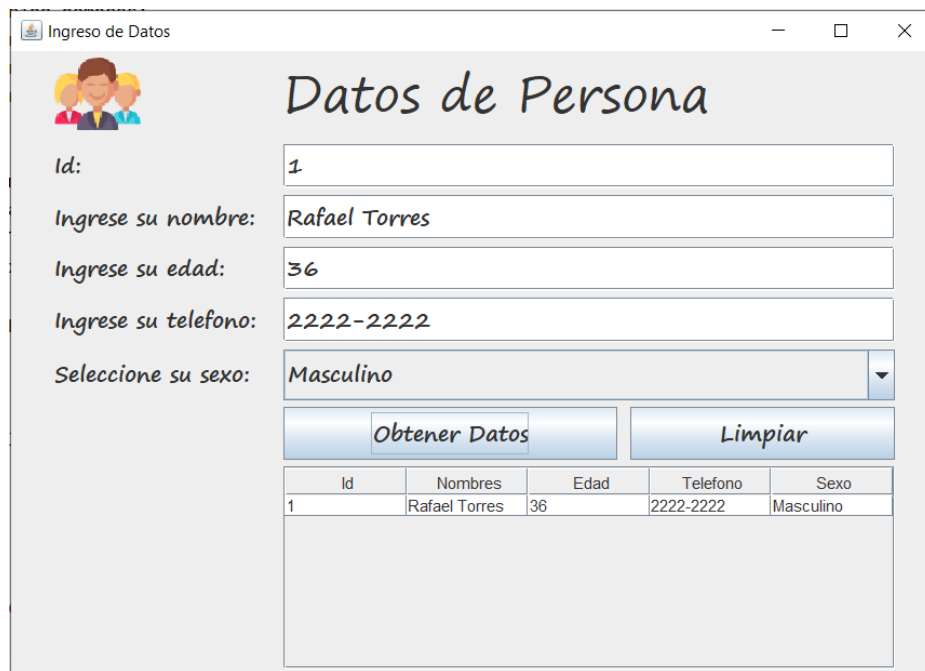
17. Para agregar los datos ingresados por el usuario, se deberá agregar el código marcado en rojo en el método btnObtenerDatos().



```
66 private void btnObtenerDatos() {
67     String id;
68     String nombres;
69     String edad;
70     String telefono;
71     String sexo;
72
73     id = txtId.getText();
74     nombres = txtNombre.getText();
75     edad = txtEdad.getText();
76     telefono = txtTelefono.getText();
77     sexo = cmbSexo.getSelectedItem().toString();
78
79     JOptionPane.showMessageDialog( parentComponent: null, message: "Datos Obtenidos: \n ID: " +id+
80         "\n nombres: " + nombres+"\n Edad: "+edad+"\n Telefono: "+telefono+
81         "\n Sexo: " + sexo);
82
83     Object[] newRow={id ,
84         nombres,
85         edad,
86         telefono,
87         sexo};
88
89     modelo.addRow(newRow);
90 }
91 }
```



18. Al ejecutar el programa, el resultado será como el que se muestra en la siguiente imagen.



**Datos de Persona**

Id: 1

Ingrese su nombre: Rafael Torres

Ingrese su edad: 36

Ingrese su telefono: 2222-2222

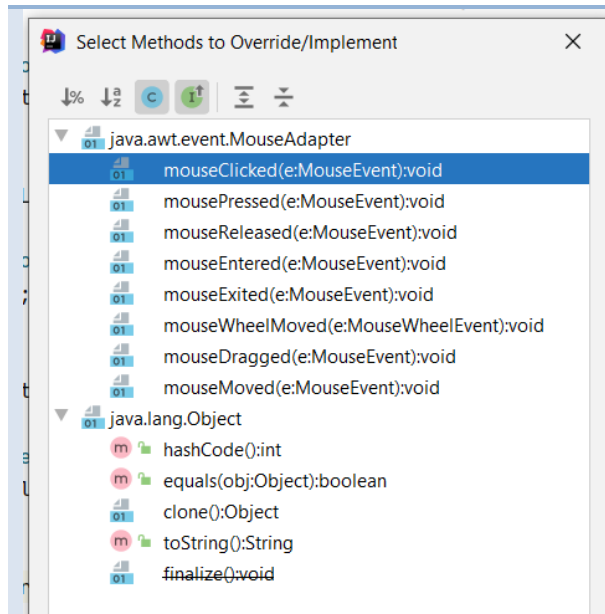
Seleccione su sexo: Masculino

Obtener Datos Limpiar

Id	Nombres	Edad	Telefono	Sexo
1	Rafael Torres	36	2222-2222	Masculino

19. Para darle un funcionamiento adicional a la tabla, se procederá a crear un evento “MouseListener” implementando el método

### Seleccionando Método



### Listener Creado en el constructor

```
tblDatos.addMouseListener(new MouseAdapter() {  
    @Override  
    public void mouseClicked(MouseEvent e) {  
        super.mouseClicked(e);  
    }  
})
```


20. Se procederá a modificar el Listener agregando un nuevo método llamado **tblObtenerDatos()**, este método contendrá el código que permitirá obtener los datos del registro seleccionado de la tabla y reflejarlos en las cajas de texto, esto con la posibilidad de que en un futuro al hacer un mantenimiento, solamente se modifiquen los campos necesarios y no se ingresen todos los datos nuevamente.

```

58     });
59     tblDatos.addMouseListener(new MouseAdapter() {
60         @Override
61         public void mouseClicked(MouseEvent e) {
62             super.mouseClicked(e);
63             tblObtenerDato(e);
64         }
65     });
66 }
67
68 @
69 private void tblObtenerDato(MouseEvent e) {
70     int fila = tblDatos.rowAtPoint(e.getPoint());
71     int columna = tblDatos.columnAtPoint(e.getPoint());
72
73     if ((fila > -1) && (columna > -1)){
74         txtId.setText(modelo.getValueAt(fila, columna: 0).toString());
75         txtNombre.setText(modelo.getValueAt(fila, columna: 1).toString());
76         txtEdad.setText(modelo.getValueAt(fila, columna: 2).toString());
77         txtTelefono.setText(modelo.getValueAt(fila, columna: 3).toString());
78         cmbSexo.setSelectedItem(modelo.getValueAt(fila, columna: 4).toString());
79     }
80 }

```

21. Al realizar la ejecución y cargar información en la tabla, se podrán seleccionar cualquier registro y verlo reflejado en los campos de texto.



## Datos de Persona

**Id:**

**Ingrese su nombre:**

**Ingrese su edad:**

**Ingrese su telefono:**

**Seleccione su sexo:**

Id	Nombres	Edad	Telefono	Sexo
1	Rafael Torres	36	2222-9285	Masculino
2	Katherinne Vigil	29	2599-9589	Femenino
3	Helen Platero	29	2589-8778	Femenino
4	Carlos Alfaro	40	2258-8589	Masculino

## Ejemplo2

Utilizaremos los RadioButton para poder ver diferentes imágenes.

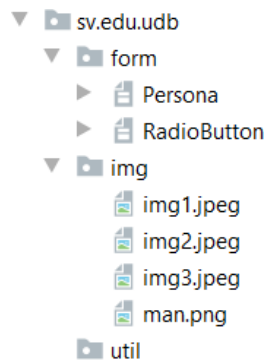
1. Agregar un nuevo JFrame con el nombre **RadioButton**.
2. Ingresar los siguientes elementos que se muestran en la imagen.



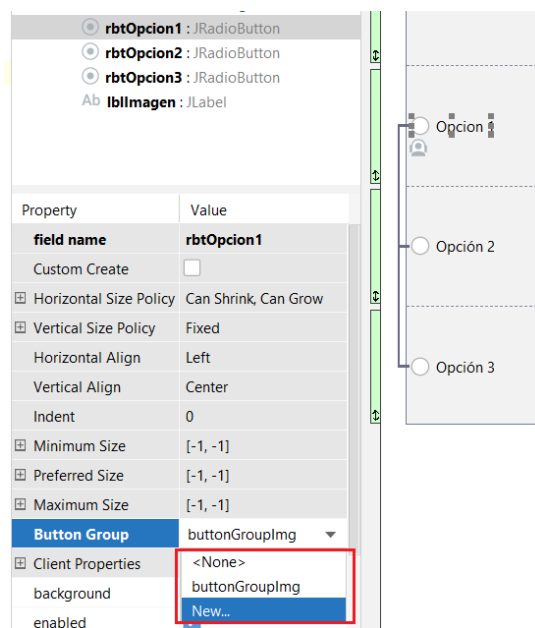
3. Modificar las propiedades según la siguiente tabla.

Control	Propiedad	Valor
<b>JLabel1</b>	text	Seleccione una imagen
	name	lblTitulo
	font	Segoe Print 22 pt Bold
<b>JLabel2</b>	text	-----
	field name	lblImagen
<b>JRadioButton1</b>	icon	sv/edu/udb/img/img1.jpeg
	text	Opcion1
	field name	rbtOpcion1
	buttonGroup	buttonGroup1
<b>JRadioButton2</b>	text	Opcion2
	field name	rbtOpcion2
	buttonGroup	buttonGroup1
<b>JRadioButton3</b>	text	Opcion3
	field name	rbtOpcion3
	buttonGroup	buttonGroup1

4. Para poder ver las imágenes las agregaremos al proyecto al paquete “sv.edu.udb.img”, estas se deben quedar como se muestran en las siguientes imágenes.



- Los RadioButton son elementos cuya funcionalidad debe permitir la selección de uno de un conjunto de RadioButtons, para ello deberá seleccionar el RadioButton buscar la propiedad **ButtonGroup** y si no existe ningún grupo deberá crear uno dando click en “New” y ponerle el nombre **buttonGroupImg**, para los otros radio botones solo deberá seleccionar para que pertenezcan a un mismo grupo.



- Agregar 3 Listener de tipo ActionPerformed para cada radiobutton, posteriormente agregar el código necesario para cambiar la imagen, a continuación, se muestra el código final de la case RadioButton.

```
package sv.edu.udb.form;

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

public class RadioButton extends JFrame{
    private JRadioButton rbtOpcion1;
    private JRadioButton rbtOpcion2;
    private JRadioButton rbtOpcion3;
    private JLabel lblImagen;
    private JPanel pnlImg;

    public RadioButton(String title) {
        super(title);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```

        this.setContentPane(pnlImg);
        this.setMinimumSize(new Dimension(400, 400));
        this.setLocationRelativeTo(getParent());

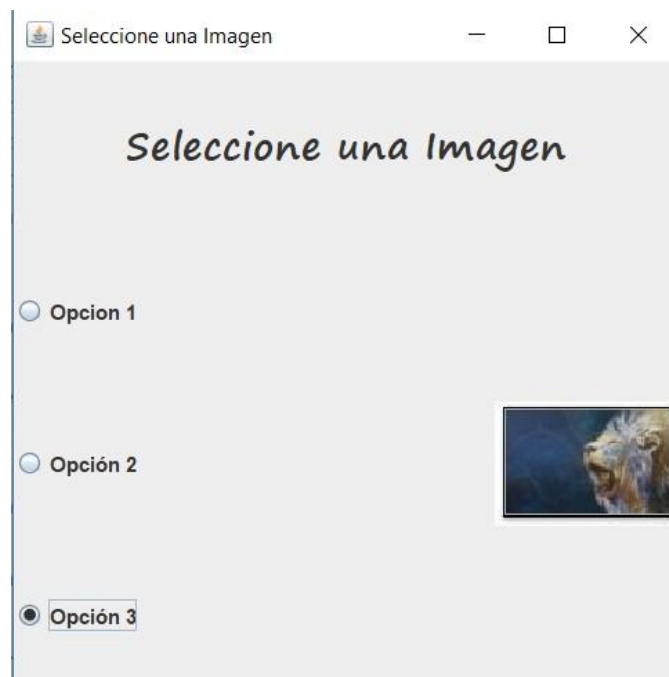
        rbtOpcion1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                lblImagen.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/sv/edu/udb/img/img2.jpeg")));
            }
        });
        rbtOpcion2.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                lblImagen.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/sv/edu/udb/img/img1.jpeg")));
            }
        });

        rbtOpcion3.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                lblImagen.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/sv/edu/udb/img/img3.jpeg")));
            }
        });
    }

    public static void main(String[] args) {
        JFrame frame = new RadioButton("Seleccione una Imagen");
        frame.setVisible(true);
    }
}

```

7. Al ejecutar el programa, este permitirá que al cambiar de opción la imagen pueda ir cambiando en el label.



## IV. EJERCICIOS COMPLEMENTARIOS

Modificar el programa del ejemplo 1, para que pueda realizar lo siguiente

1. Leer la información de un archivo csv, y al cargar el programa la tabla pueda aparecer llena para seleccionar los datos.  
Id,nombres,edad,teléfono,sexo  
1,Rafael Torres,36,2222-5689,Masculino  
2,Carlos Alvaro,42,24598-6587,Masculino  
3,Karens Medrano,35,2655-5686,Femenino
2. Escribir en el archivo, la información que es obtenido por medio del formulario, esta información debe añadirse al final del último registro que este contenga.

### Resultado personal:

Id	Nombre	Edad	Telefono	Sexo
2	Pedro	18	2432345645	Femenino
3	Pablo	20	21348734789324	Femenino
4	Sofia	8	243244543	Femenino
4	Safiro	30	78907645	Masculino

2	Pedro	18	2432345645	Femenino
3	Pablo	20	21348734789324	Femenino
4	Sofia	8	243244543	Femenino
4	Safiro	30	78907645	Masculino

## V. BIBLIOGRAFIA

- [http://www.javahispano.org/contenidos/archivo/63/jtable\\_1.pdf](http://www.javahispano.org/contenidos/archivo/63/jtable_1.pdf)

Última fecha de visita: 11/02/2023

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

Última fecha de visita: 11/02/2023