**Java 28 – 2019 JSE EIT – BUI**



**Java en forma visual: NetBeans IDE GUI builder**

Este apunte es una introducción al **NetBeans IDE GUI builder**, que nos genera programas usando la **GUI** (**G**raphical **U**ser **I**nterface), la biblioteca **Swing** y el **IDE** **NetBeans**.

Le vamos a dar un nombre más fácil a este tipo de programación, y lo llamaremos “**Java en forma visual**”, que tiene sus ventajas y sus desventajas con respecto al uso de los **Layouts**, tema que veremos más adelante.

Ventajas:

* Es fácil de ubicar y colocar propiedades a los controles.
* Se trabaja más rápido.

Desventajas:

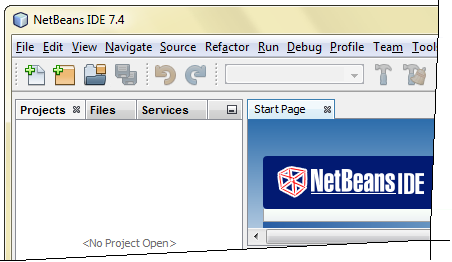
* El programa que nos genera no siempre es el que podemos entender.
* Parte del código es inmodificable por nosotros.
* Como todo lenguaje que genera código, a veces coloca instrucciones de más.
* Esto está muy orientado al **IDE** con el que trabajamos, lo que significa por ejemplo, si estamos trabajando con el **NetBeans** en forma visual, mejor no cambiemos de **IDE**.

Finalmente diré que está en nosotros elegir nuestra forma de trabajo, pero personalmente recomendaré el uso de la forma visual.

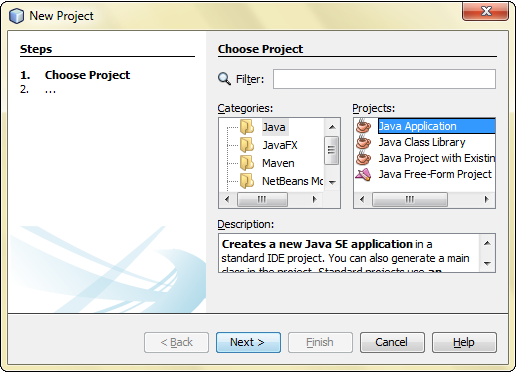
**Primer programa en forma visual**

Comenzamos de cero.

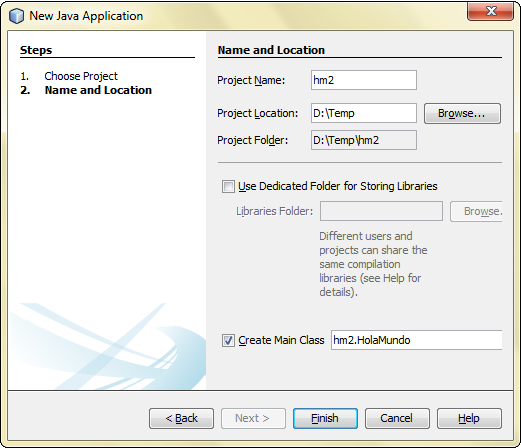
Seleccionamos una nueva aplicación (**New Project**).



Entonces nos aparece el cuadro de diálogo “**New** **Project**”, en la que seleccionamos “**Java**”, “**Java Application**” y “**Next >**”.



En el cuadro de diálogo “**New Java Application**”, colocamos el nombre del proyecto (**Project Name**), en donde se lo ubica (**Project Location**) y la Clase principal (**Create Main Class**).



Nos aparece un programa.

Le sacamos casi todos los comentarios y nos queda el siguiente programa:

package hm2;

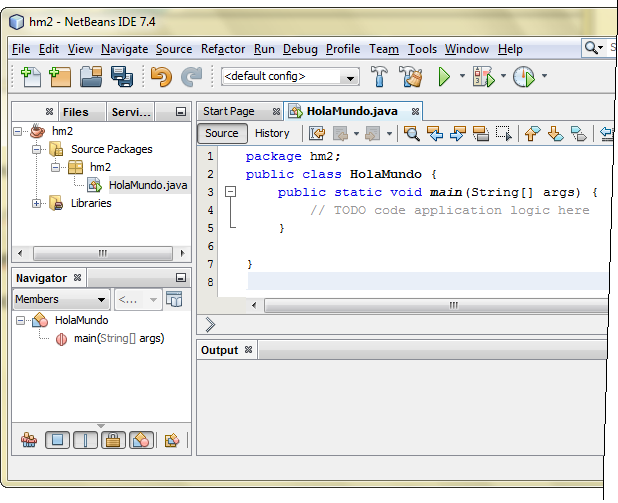
public class HolaMundo {

public static void *main*(String[] args) {

// TODO code application logic here

}

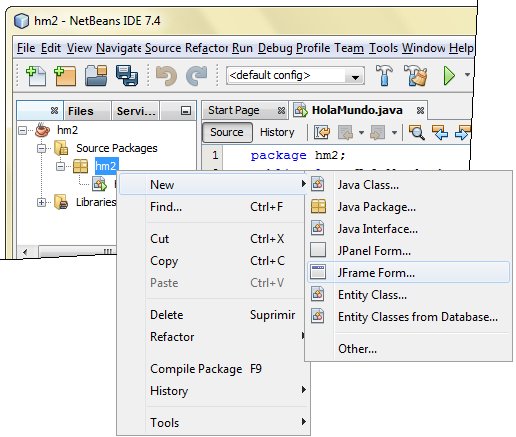
}



Ahora comencemos con la parte visual.

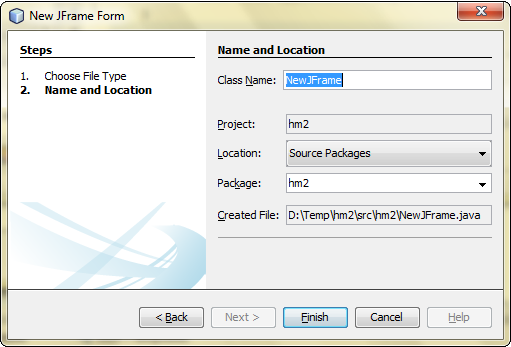
Presionamos con el botón derecho del mouse al paquete “**hm2**”…

…y nos aparece un menú desplegable, en el que seleccionamos “**New**” y luego “**JFrame Form…**”.

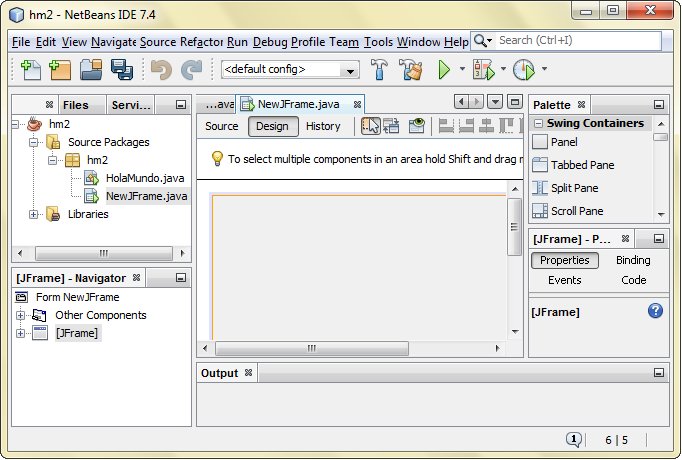


Entonces, nos aparece un cuadro de diálogos llamado “**New JFrame Form**”.

Podemos ponerle nombre a nuestra clase (**Class Name**), sin embargo, dejamos el nombre que nos propone (**New JFrame**), y presionamos el botón “**Finish**”.



Nos aparece un formulario (“**Frame**”) vacío.

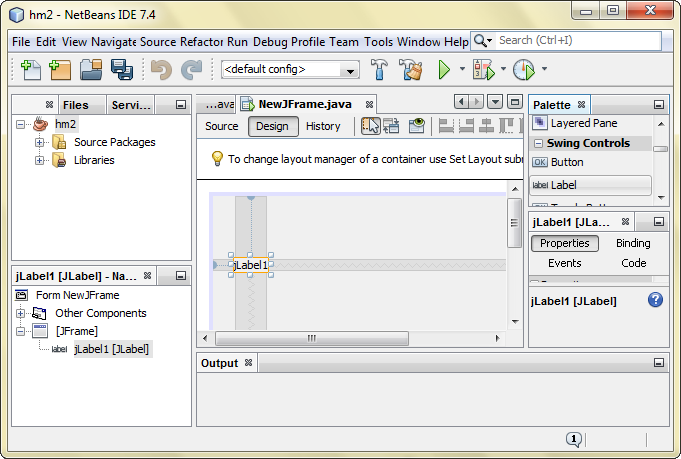


Le agregamos a este frame los controles que desamos, por ejemplo de la subventana “**Palette**”…

…del sector “**Swing Controls**”…

…seleccionamos el ítem “**Label**”…

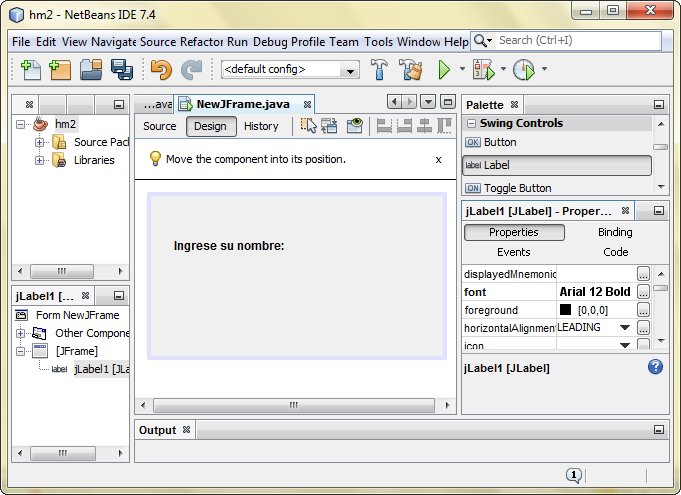
…y lo instanciamos en el frame (flecha azul).



Teniendo seleccionado ese control, que se llama “**jLabel1**”, con la subventana “**Properties**” (debe estar seleccionado ese botón)…

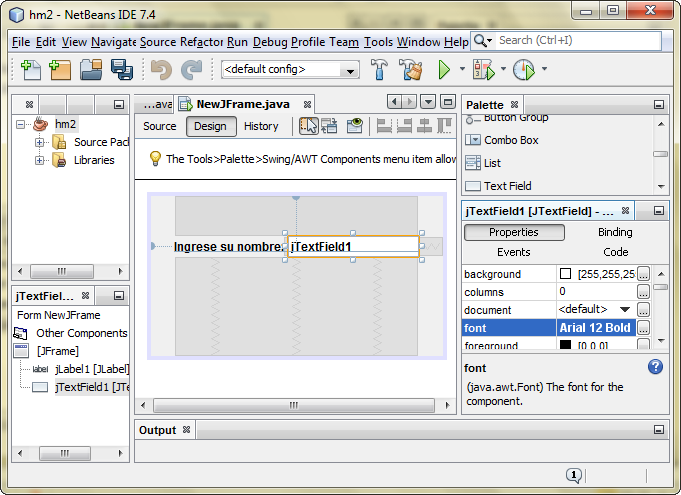
…cambiamos las propiedades:

* “**text**”: “Ingrese su nombre”.
* “**font**”: “Arial 12 Bold”.

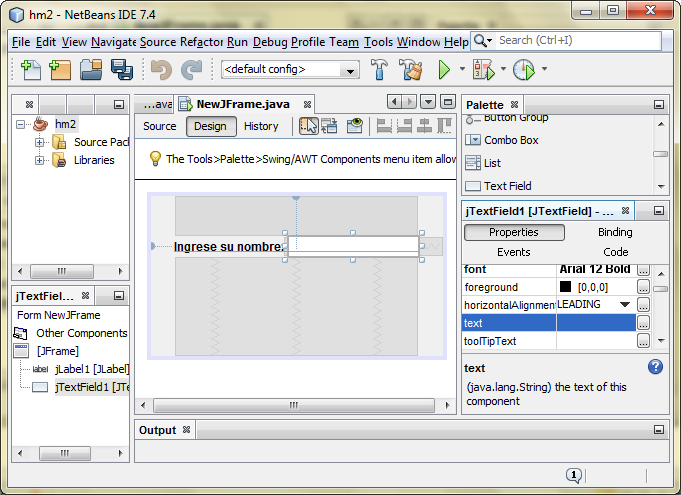


Agregamos al frame un control “**TextField**” (se llama “**jTextField1**”) y le cambiamos las propiedades…

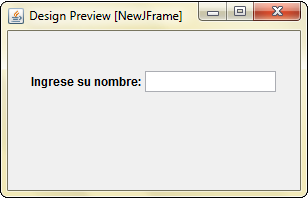
* “**font**”: “Arial 12 Bold”.



* “**text**”: (lo dejamos en blanco).



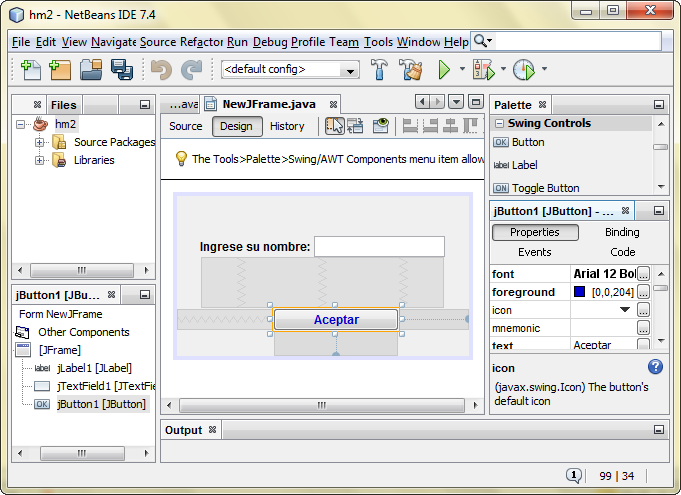
Si presionamos el botón de vista previa (“**Preview Design**”) , podemos ver como nos va a quedar el frame cuando ejecutemos nuestro programa:

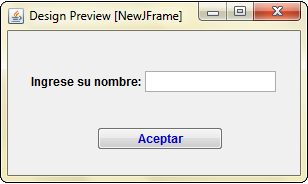


Importante: esta no es la ejecución del programa, sino una vista previa del **frame**.

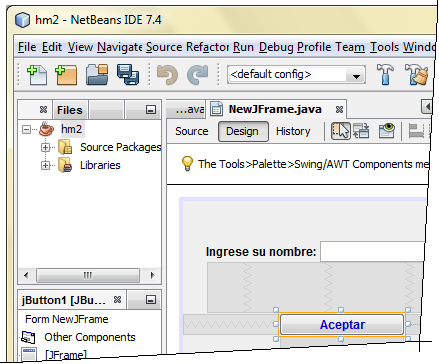
Le agregamos al frame un control “**Button**” (se llama “**jButton1**”) y le cambiamos las propiedades…

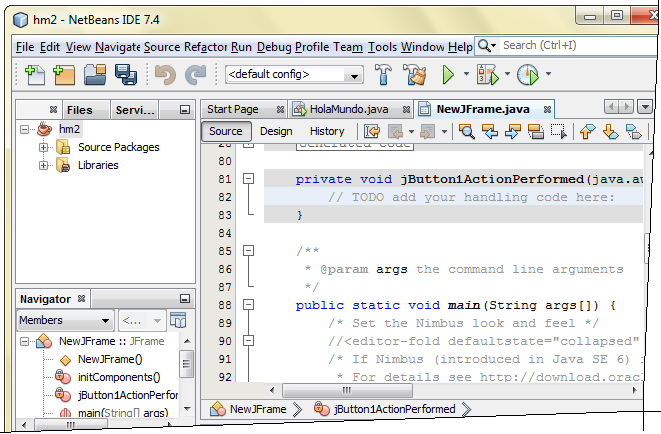
* “**font**”: “Arial 12 Bold”.
* “**foreground**”: Nos muestra un **JColorChooser**, en donde seleccionamos un color de **RGB** de [0,0,204].
* “**text**”: “Aceptar”.



Vemos la vista previa con el botón ya incorporado:

Al hacer doble click sobre el botón “**Aceptar**” en el modo “**Design**”, el **IDE** se muestra en el modo “**Source**” (programa fuente).

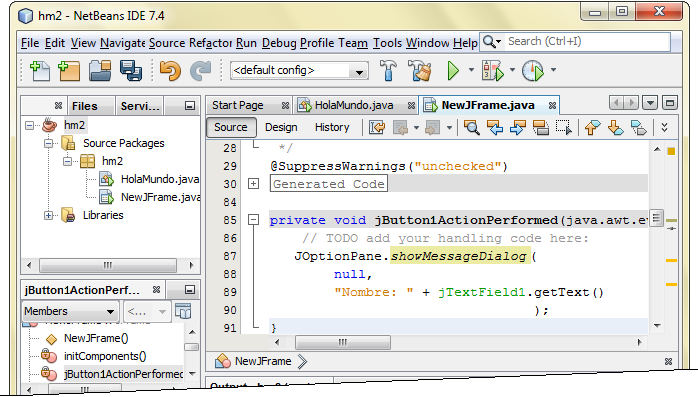




En este lugar…

escribiremos código de programa.

Agregamos como código de programa un “**JOptionPane.showMessageDialog**”.



JOptionPane.showMessageDialog(

null,

"Nombre: " + jTextField1.getText()

);

Si esta instrucción nos da error, seguramente no tenemos especificada su biblioteca.

Agregamos después del paquete, la siguiente instrucción:

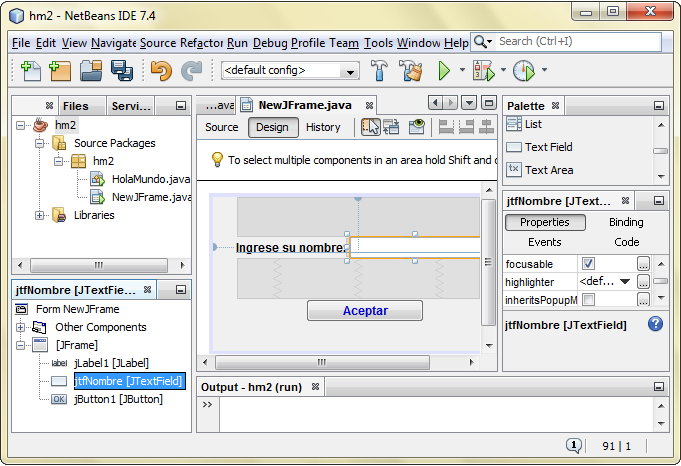
import javax.swing.\*;

Deseamos cambiarle el nombre al **jTextField1**.

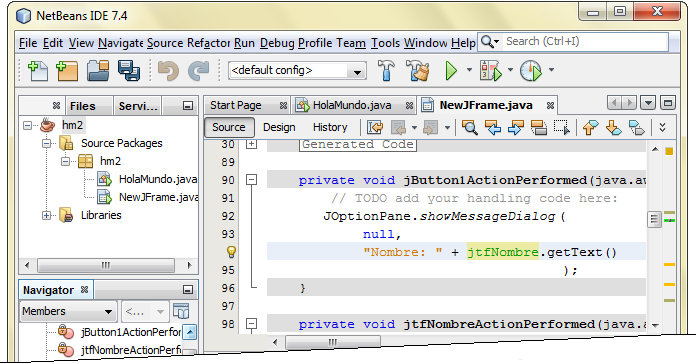
En el modo “**Design**”, seleccionamos el control que deseamos cambiarle el nombre.

Entonces, le hacemos un doble click pausado (no muy seguido) y nos habilita para hacer el cambio.

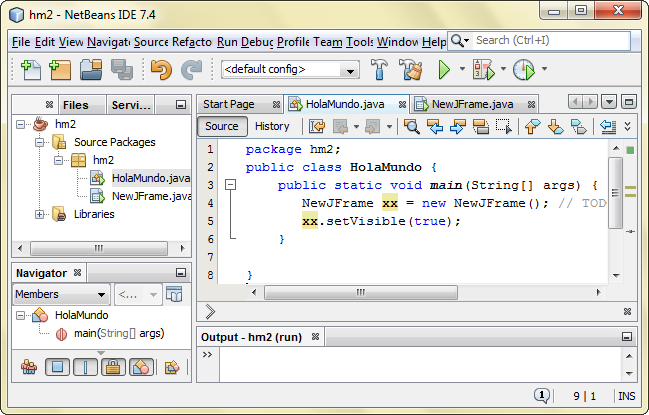
En este ejemplo, hemos cambiado el nombre **jTextField1** por **jtfNombre**.



Lo interesante de este cambio, es que se producen cambios automáticos dentro del programa, relacionados con este nombre.



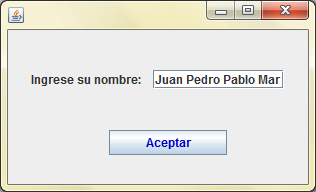
Desde el archivo “**HolaMundo.java**”, creamos el objeto “**xx**” y lo hacemos visible.



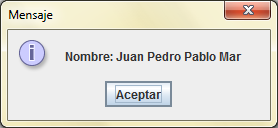
NewJFrame xx = new NewJFrame(); // TODO code application logic here

xx.setVisible(true);

La salida que nos brinda este programa es:



Ingresamos un nombre, presionamos el botón “**Aceptar**” y nos muestra este mismo nombre en una ventana de mensajes:

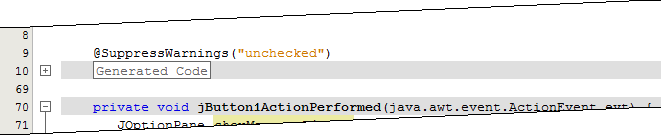


Las partes del programa que aparecen **sombreadas en gris** corresponden al diseño visual que se generó automáticamente y que **no puede modificarse** con el editor.

Incluso hay alguna sección en gris que aparece oculta bajo el texto "**Generated Code**", que puede verse desplegando el signo “**+**” que aparece junto a él.



Entonces, desplegando el signo “**+**”, vemos las filas ocultas, las que van de la fila 11 a la 68.



…expandimos y vemos el código de programa oculto.



En vez de haber escrito la visibilidad en la clase “**HolaMundo**”:

package hm2;

public class **HolaMundo** {

public static void main(String[] args) {

NewJFrame xx = new NewJFrame();

**xx.setVisible(true);**

}

}

Pudimos haberlo hecho en el constructor de la clase “**NewJFrame**”:

package hm2;

import javax.swing.JOptionPane;

public class **NewJFrame** extends javax.swing.JFrame {

public NewJFrame() {

initComponents();

**setVisible(true);**

}

…

}

Podemos agregarle a este constructor alguna otra propiedad como por ejemplo:

setLocation( 300, 100 );

setTitle( "Primer programa visual" );