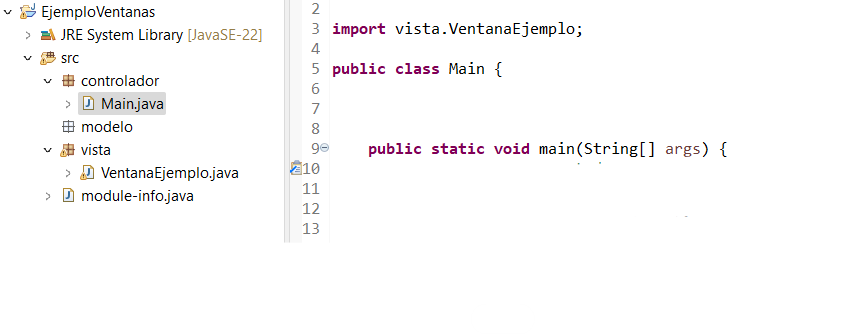
Tras instalar la extensión windowbuilder en Eclipse (help > marketplace > search > windowbuilder current), instalamos.

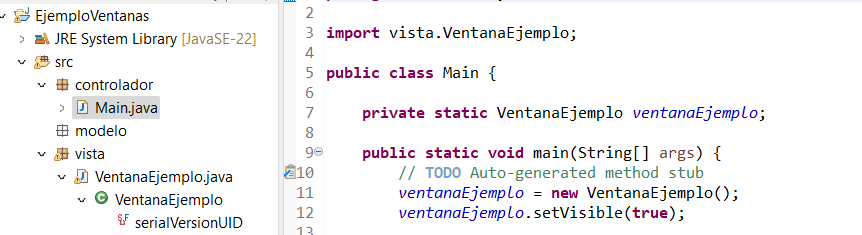
Creamos un nuevo proyecto en Java con 3 paquetes: modelo (de momento vacío), vista (contiene toda la información de las ventanas) y controlador (contiene un archivo Main.java, con su clase public void string…).



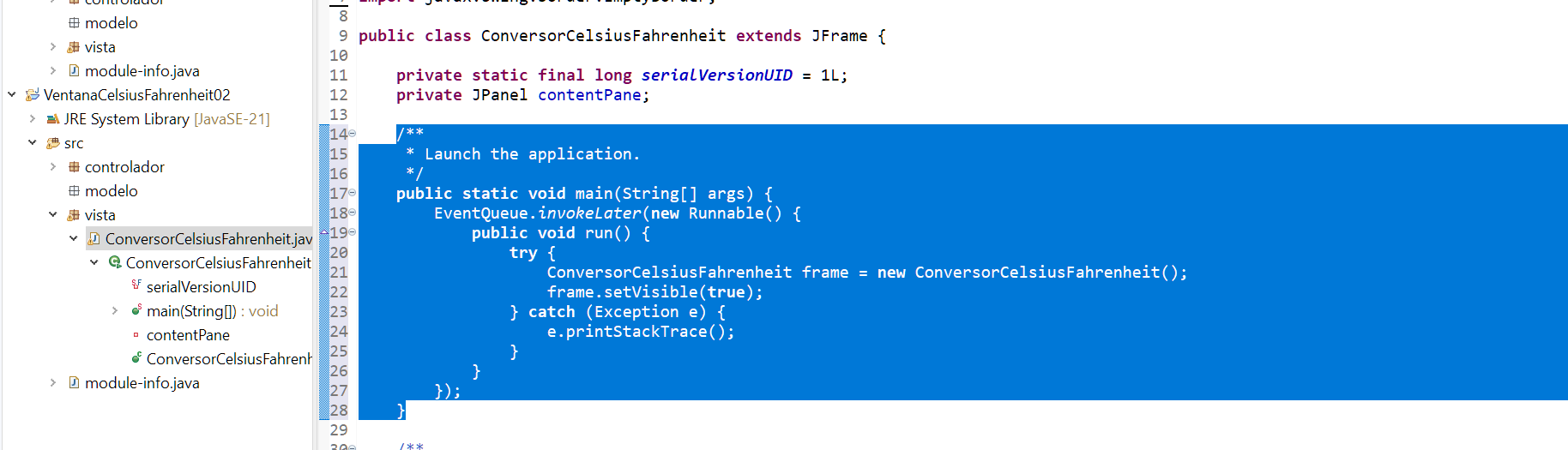
En el paquete de vista creamos un nuevo archivo: new > other > windowbuilder > Swing Designer > JFrame. Este archivo lo nombraremos como nuestra interfaz gráfica. Click en Design (está en la zona de la consola).



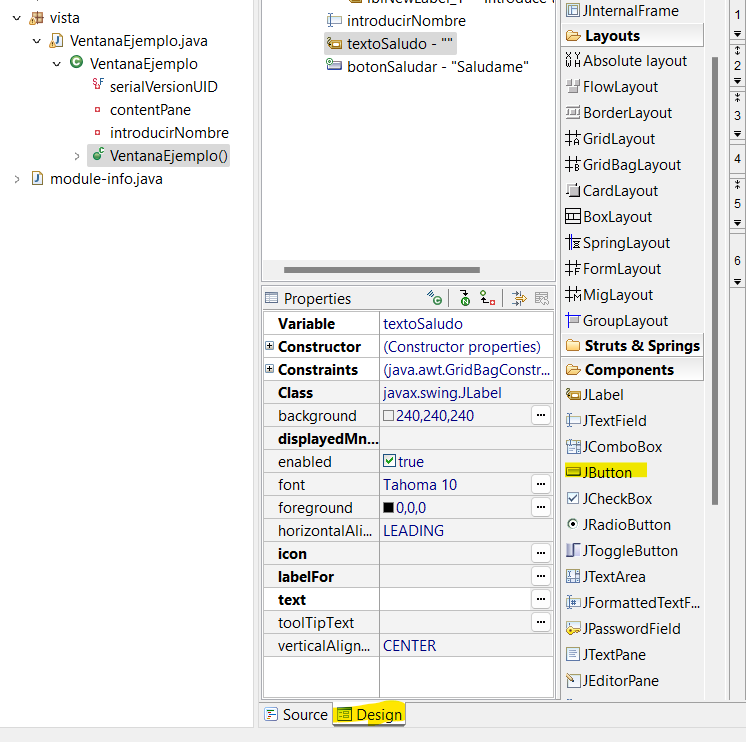
En el archivo main del paquete controlador creamos una clase private y a continuación la instanciamos, además de darle visibilidad (tendremos que importar el archivo de la vista, Eclipse lo pide).



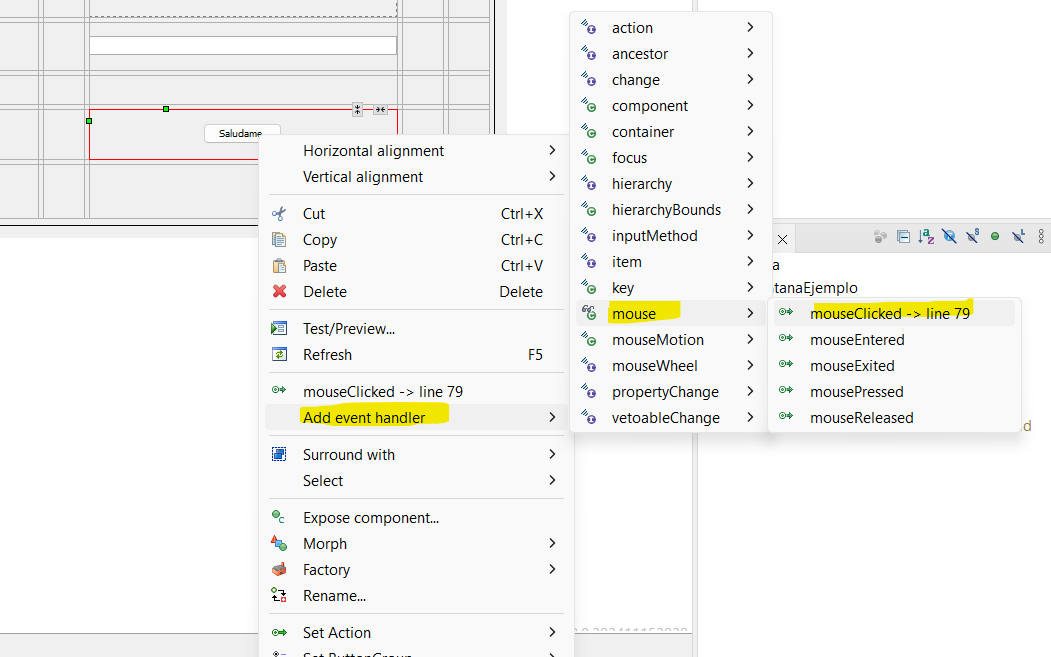
En el archivo del paquete vista eliminamos toda la parte del main (el bloque the launch the application) porque solo puecde haber un main, el del modelo controlador.



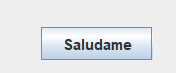
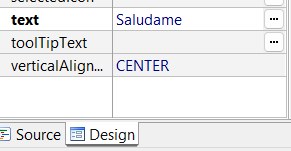
Vamos al archivo VentanaEjemplo, a la pestaña Design, y buscamos en la lista un JButton.



Arrastramos el JButton hacia la ventana de la derecha. Una vez ahí click derecho, add event listener > mouse > mouse clicked (para indicar que se hará algo cuando cliquemos ahí.



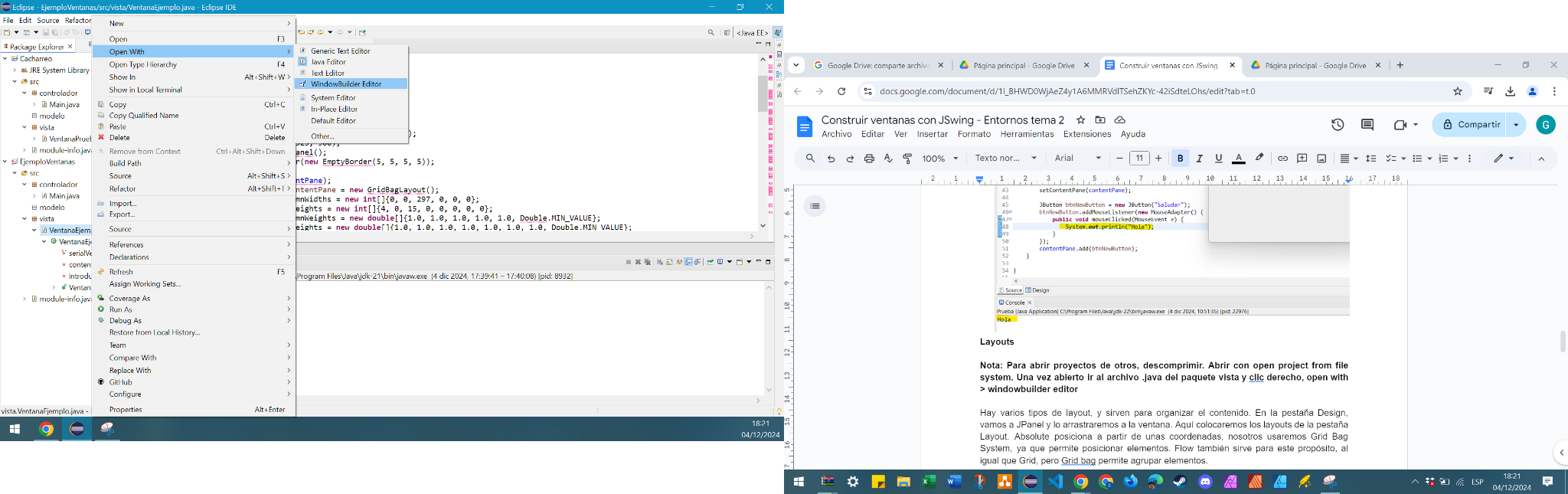
Para cambiar el texto del botón vamos a la pestaña Design de nuevo y en el panel de la izquierda cambiamos el valor de text.



Para hacer que el botón haga algo, nos vamos a source y en las líneas de código que se han generado ponemos un system out:



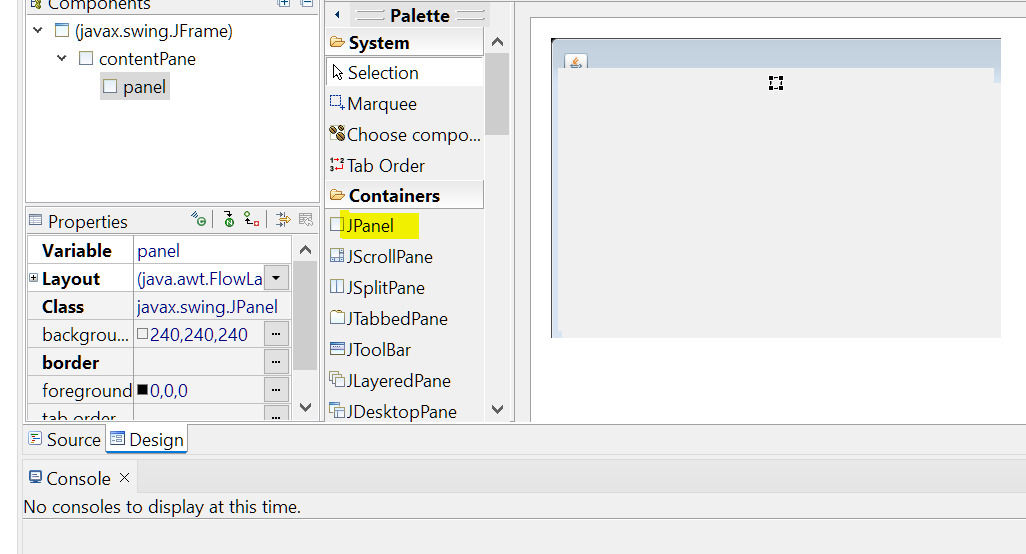
**Nota: Para abrir proyectos de otros, descomprimir. Abrir con open project from file system. Una vez abierto ir al archivo .java del paquete vista y clic derecho, open with > windowbuilder editor**

****

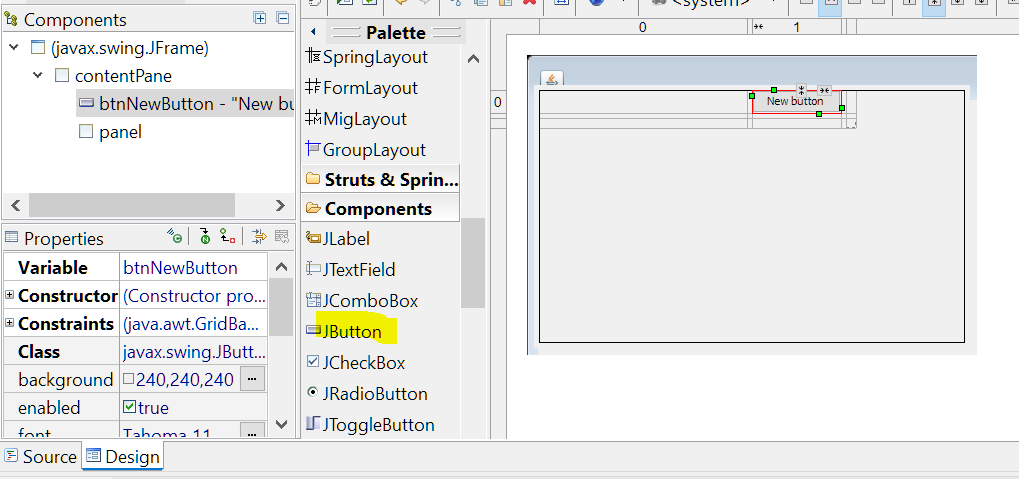
**Layouts**

Hay varios tipos de layout, y sirven para organizar el contenido. En la pestaña Design, vamos a JPanel y lo arrastraremos a la ventana. Aquí colocaremos los layouts de la pestaña Layout. Absolute posiciona a partir de unas coordenadas, nosotros usaremos Grid Bag System, ya que permite posicionar elementos. Flow también sirve para este propósito, al igual que Grid, pero Grid bag permite agrupar elementos.

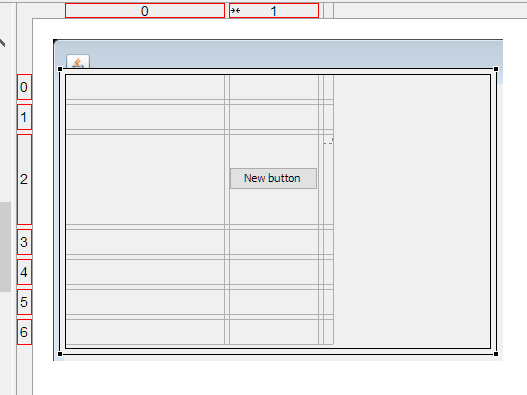
Primeramente colocamos un JPanel dentro.



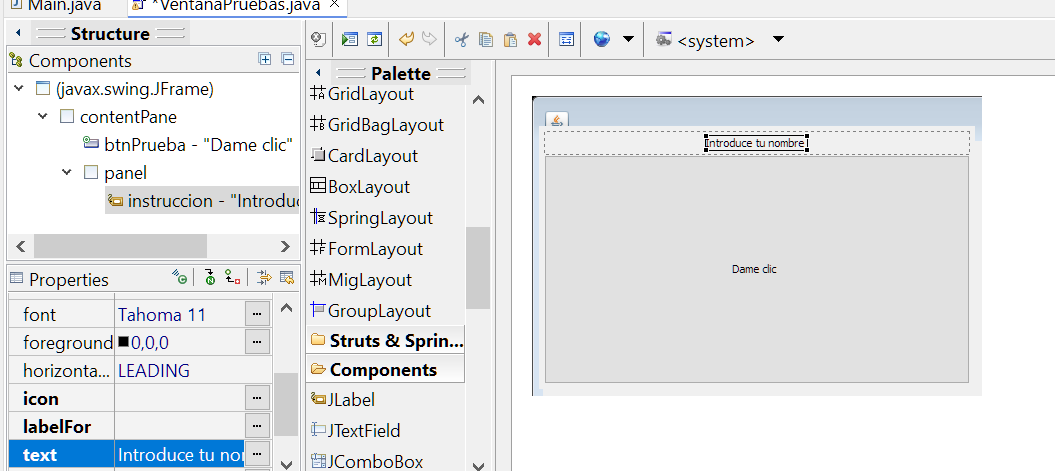
A continuación colocamos el Grid Bag, y dentro un botón.



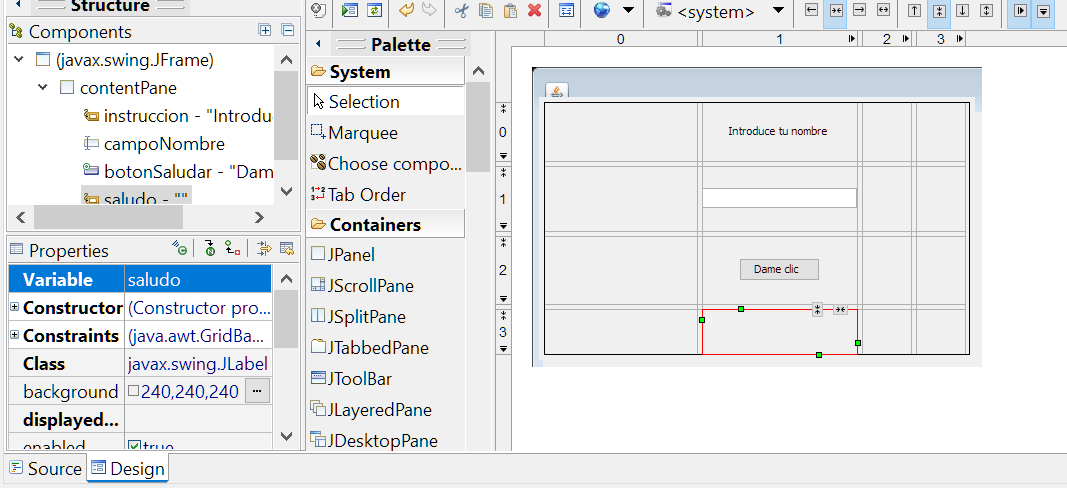
A continuación seleccionamos las filas y las columnas (los números de las esquinas), clic derecho y le damos a grow. Así, el programa cogerá todo el ancho y alto disponible y el botón siempre estará en el centro.



A continuación metemos un JLabel, que sirve para mostrar un texto en la ventana, pero antes de eso colocaremos un BorderLayout. Esto nos permitirá colocar el JLabel en la posición que queramos.



Ahora colocamos un JTextField, que equivale a un input de tipo texto, y también un botón. Colocaremos otra JLabel bajo el botón.



En la pestaña source vamos al trozo de código que hace que el botón reaccione y copiamos lo siguiente:

JButton botonSaludar = **new** JButton("Dame clic");

botonSaludar.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

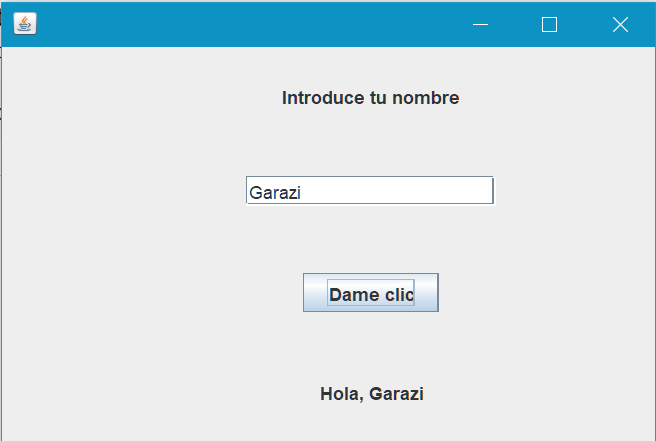
String nombre = campoNombre.getText();

saludo.setText("Hola, " + nombre);

}

});

De esta manera, guardamos en una variable lo que captura el teclado, y posteriormente hacemos que en el JLabel que está bajo el botón creemos el mensaje Hola + el nombre que hayamos introducido.



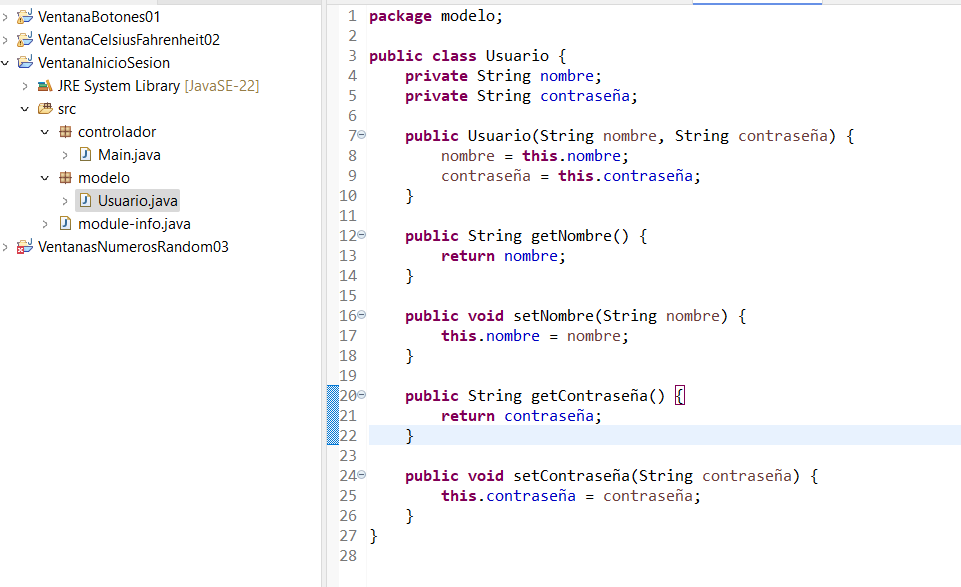
Para exportar el código lo hacemos como siempre, export > general archive file. Si lo que queremos es solamente el código ejecutable vamos a File > Export > Java > Runable JAR file > guardamos con formato **.jar**.

Para cerrar la ventana al hacer click con un botón basta con ir al código y escribir dispose();

**Cambiar de ventanas - Proyecto inicio sesión**

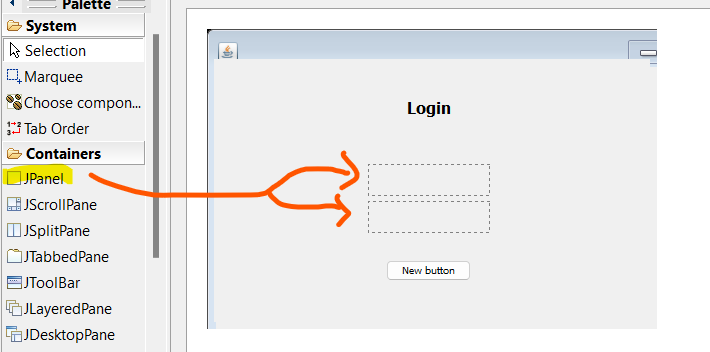
Vamos a crear un nuevo proyecto de inicio de sesión. Si las credenciales son correctas se abrirá una nueva ventana.

Creamos los tres paquetes como siempre, en el paquete modelo creamos una clase llamada Usuario.

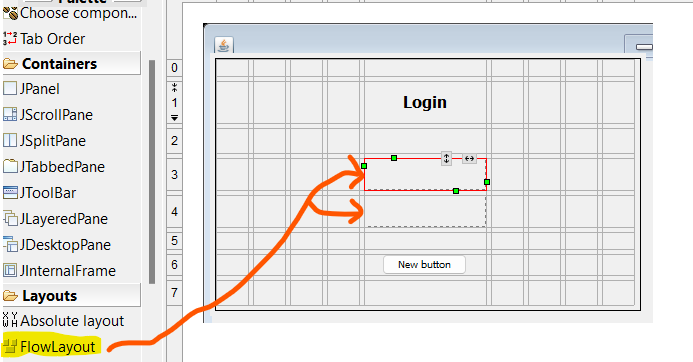


Nos vamos al paquete vista y creamos un new > other > JFrame para un archivo llamado VentanaLogin y para otro llamado VentanaLogin.

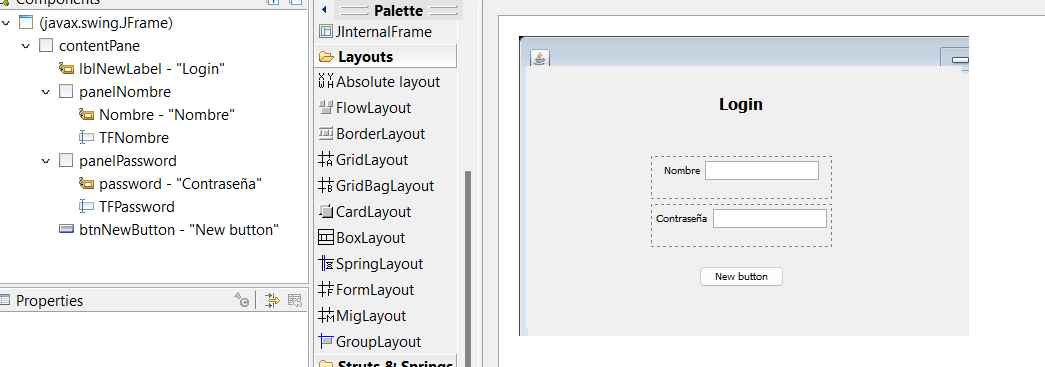
En la VentanaLogin vamos a Design y creamos una nueva ventana, metiendo un GridBagLayout y dándole a Grow. Metemos un componente llamado JPanel dos veces. Esto hará las veces de contenedor.



Dentro de estos JPanel meteremos un FlowLayout. Esto permitirá alinear horizontalmente dos elementos (JLabel y JTextField en este caso).

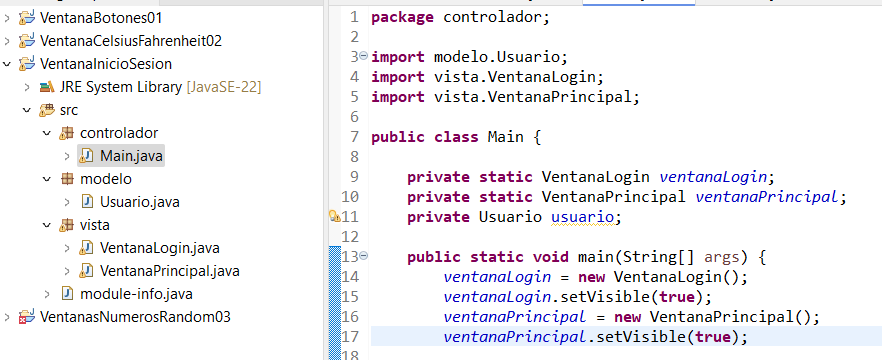


Y queda así:



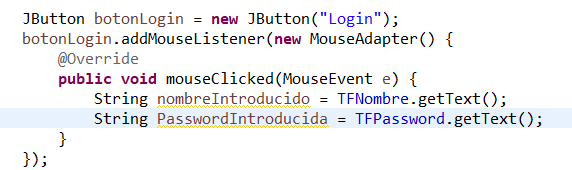
Se pueden cambiar las propiedades en el panel de properties, a la izquierda. Seleccionar el panel cambiará las propiedades de todos los elementos que están dentro, y seleccionar un elemento permitirá cambiar solamente dicho elemento.

En la pestaña Source del controlador crearemos las clases de las ventanas. Como van a operar con muchos objetos, serán estáticas. También crearemos en Main una instancia de la clase Usuario, que importaremos del paquete modelo.



Al pulsar el botón de Login en VentanaLogin cambiaremos de ventana, pero solamente si el usuario y la contraseña son correctos. Para esto, antes deben compararse con los datos disponibles.

En la pestaña Design de VentanaLogin añadimos un event handler de ratón de tipo hacer clic, y en su código correspondiente vamos a capturar los valores de los campos de texto.

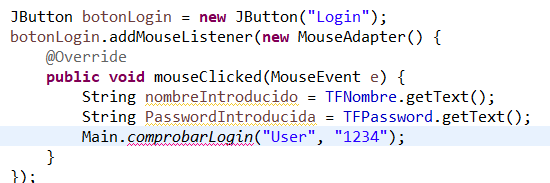


Haremos la comparación de los datos en el Main del paquete controlador.

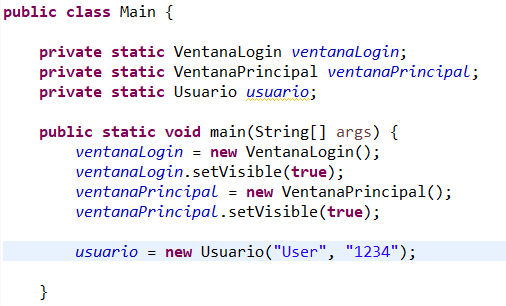
**public** **static** **void** comprobarLogin(String nombreIntroducido, String passwordIntroducida) {

}

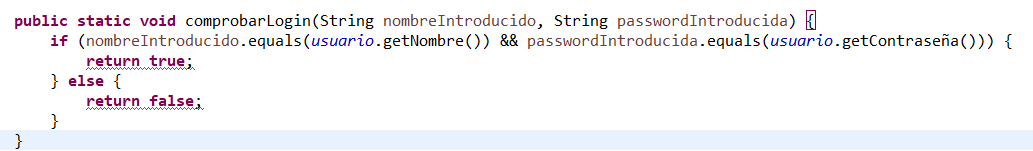
Llamaremos a esta función desde el código de la ventanaLogin, pero dará error porque hay que llamar a la función Main.



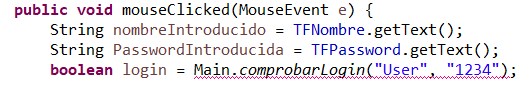
Esto compara lo que tenemos en los campos con los valores introducidos, pero antes hay que crear el usuario en el Main (recordemos que la clase en sí está creada en el modelo, y que la hemos importado a Main):



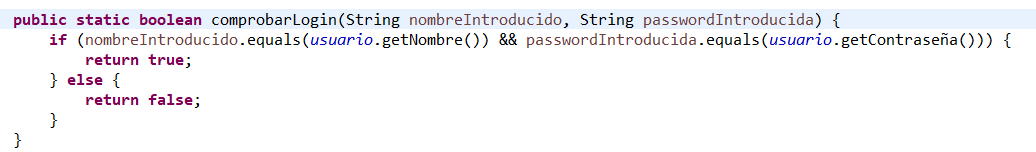
Ahora podemos rellenar la función de comprobación:



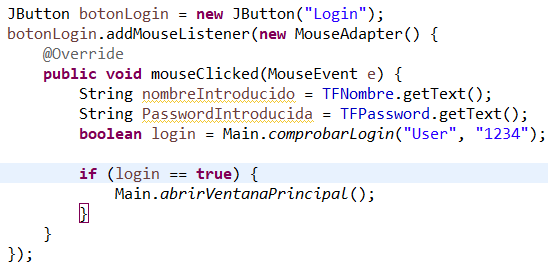
Creamos una variable booleana en el código de la ventana para guardar la llamada a la función de comprobación:



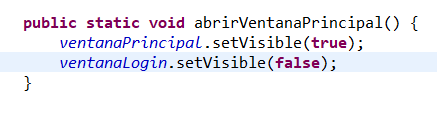
Y nos dice que debemos cambiar el tipo de dicha función a booleano.



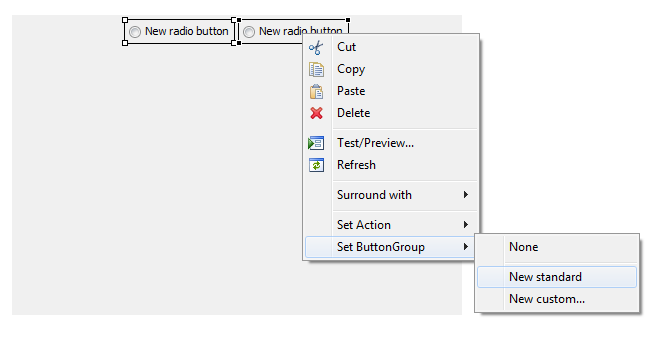
Ahora, en el código de la ventanaLogin haremos que cuando la función de comprobación devuelva true abra la otra ventana y cierre la de login (crearemos la función en el Main enseguida):

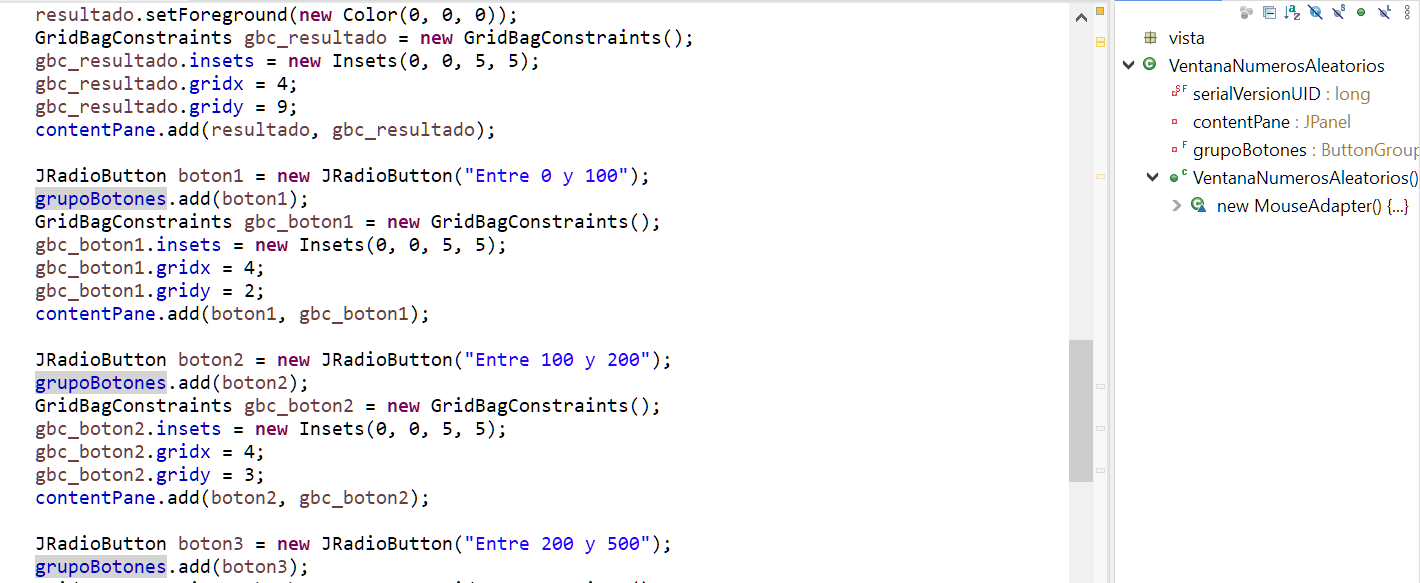


Ahora creamos abrirVentanaPrincipal() en el Main:



**Agrupar botones:** Para agrupar botones y que solo se seleccione uno a la vez, seleccionarlos todos, click derecho, seleccionar set Button Group.



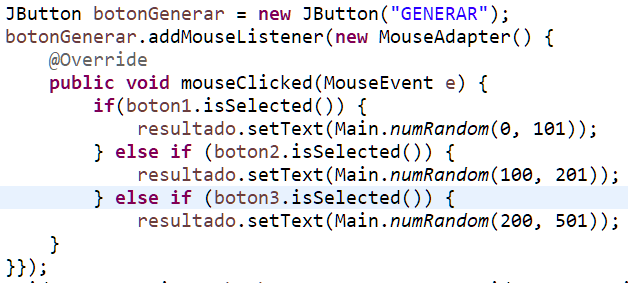


Eclipse los agrupará automáticamente.

**Hacer que en función del botón seleccionado se genere una cosa u otra**: usar condicionales con

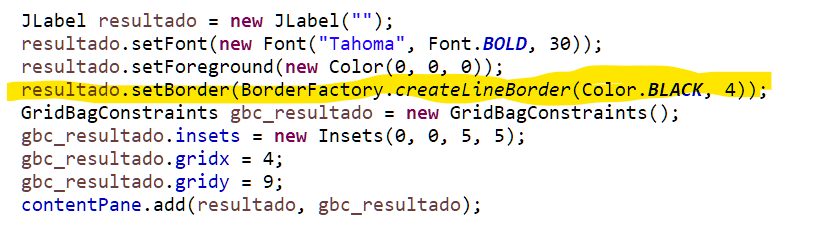
if (elemento.isSelected() {

ejecutar lógica}



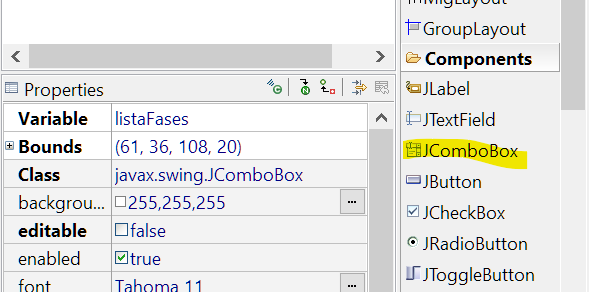
**Poner borders a las JLabels**: Vamos a la pestaña Source y buscamos el código que almacena el número. A esto le aplicaremos un BorderFactory, primero importamos la librería correspondiente:

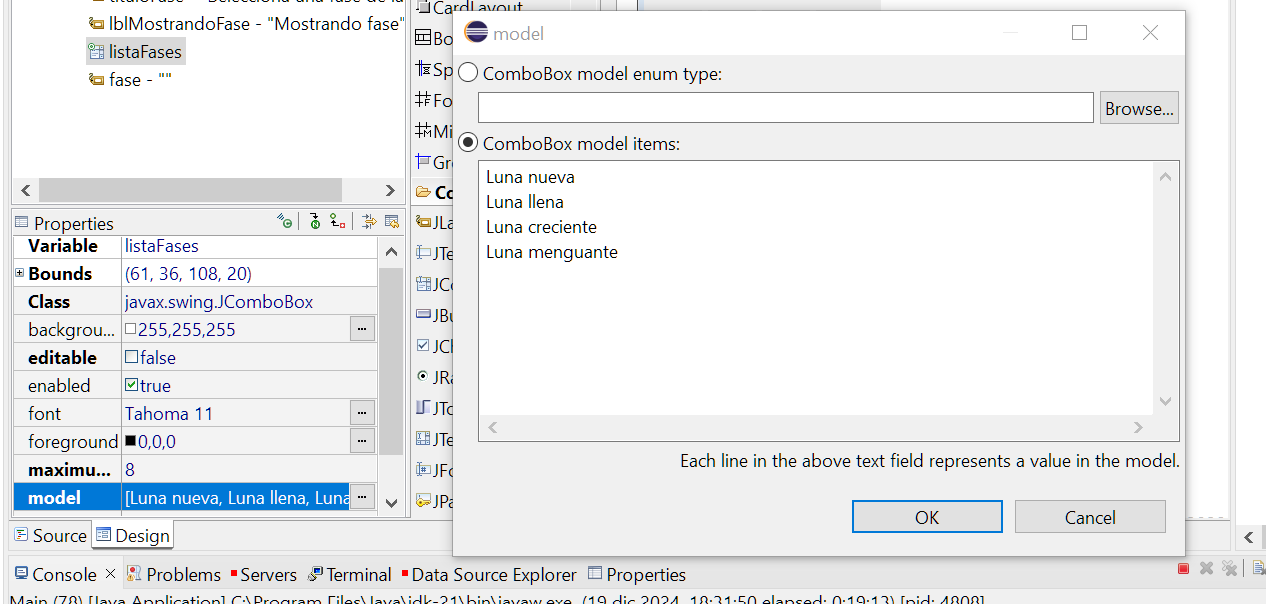
**import** javax.swing.BorderFactory;



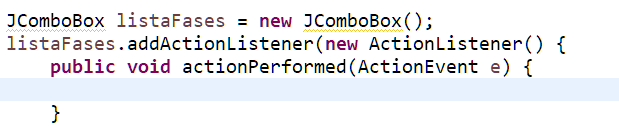
**Hacer que el programa responda a una opción de una lista**:

Seleccionamos un componente llamado ComboBox y metemos las opciones:





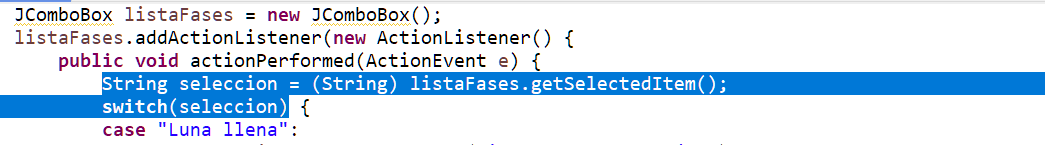
Añadimos un action handler. Esto permitirá que al seleccionar una opción se muestre la imagen correspondiente.



**Crear carpeta para guardar imágenes (tip general de Eclipse): Click derecho al proyecto > new > source folder**

**Poner imágenes en JLabels**

Tras guardar las imágenes correspondientes en la carpeta de imágenes vamos a usar un evento de tipo action, y a guardar en una variable el item seleccionado:



Y a continuación usaremos esa variable para generar un switch que cambie de imagen según nuestra selección. A continuación creamos un objeto de tipo ImageIcon y lo instanciamos, pasándole la ruta de la imagen:



Creamos otro objeto del mismo tipo que nos servirá para adaptar la imagen al tamaño del JLabel:



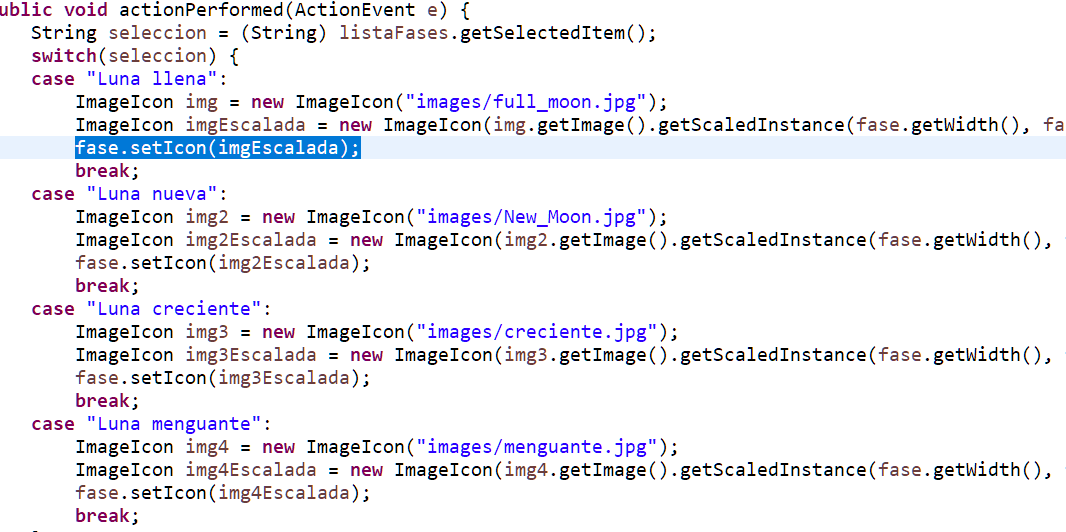
A este segundo objeto le pasamos el siguiente código:

ImageIcon imgEscalada = **new** ImageIcon(img.getImage().getScaledInstance(fase.getWidth(), fase.getHeight(), Image.***SCALE\_SMOOTH***));

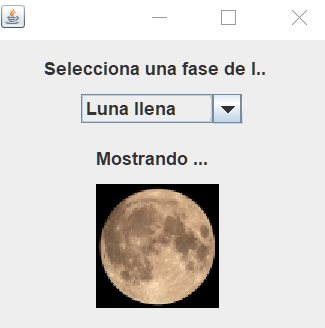
Al primer objeto que hemos creado y que contiene la ruta de la imagen vamos a pasarle getImage(=, y a esto getScaledInstance. Especificamos ancho, alto y el tipo de escalado que queremos.

Finalmente solo nos queda poner el icono en nuestro JLabel y poner el break para poner el resto de casos, cambiando la ruta y los nombres de los objetos para que no haya duplicados.



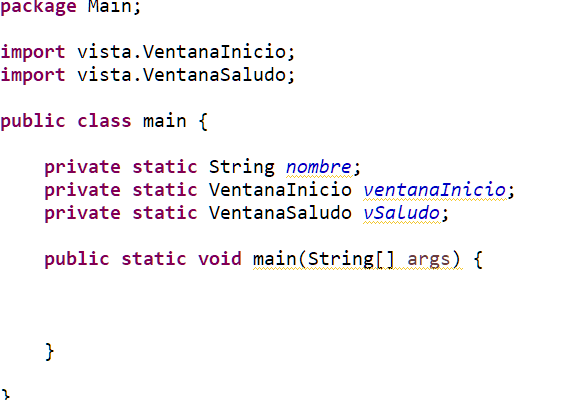


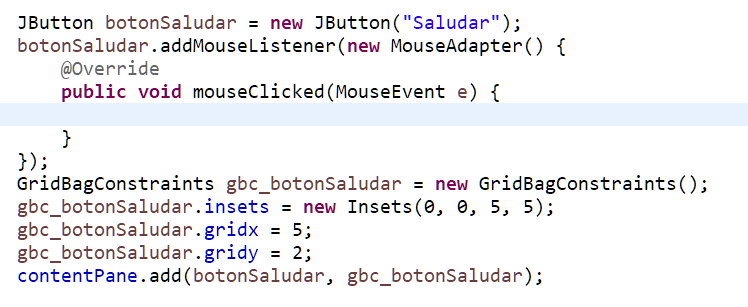
Y queda así:



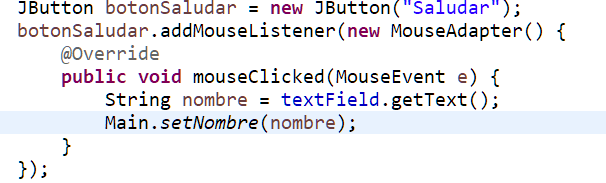
**Pasar un dato de una ventana a otra**

Creamos dos nuevas ventanas. En una de ellas, VentanaInicio, diseñamos la interfaz, compuesta por un cuadro de texto y un botón. Al botón le añadimos un evento de mouse. Es importante añadir JPanels para las etiquetas, para que funcione.



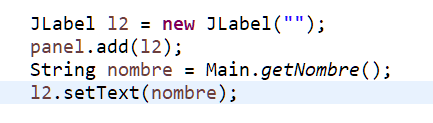


Vamos a guardar el nombre en una variable para luego llevarlo a Main. Necesitaremos getters y setters (están en Main).



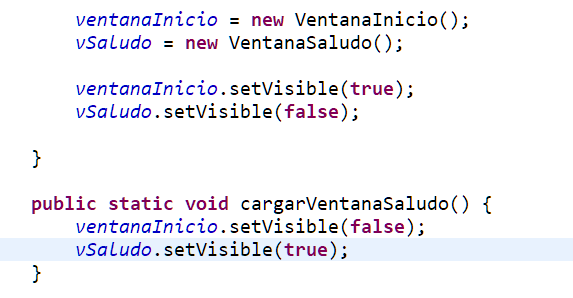
Ahora vamos a diseñar la ventanaSaludo. En un jPanel meteremos 2 jLabel, uno en el que ponga Bienvenido, y en el otro el nombre.

En esta segunda, escribimos lo siguiente:

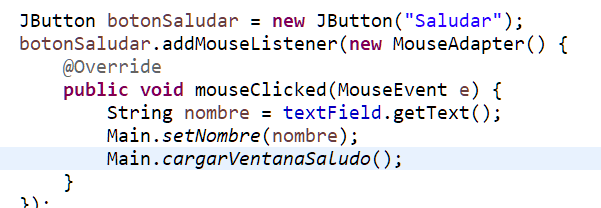


Recogemos el nombre de Main y lo pasamos al segundo jlabel.

Ahora queda poner una ventana sin visibilidad y la otra con, todo esto irá dentro de una función.



Ahora llamamos a esta función desde el evento del botón:



Vamos a obtener null al meter los datos. Para solucionarlo hay que ir al código de las JLabel y actualizarlas directamente:

JLabel tagCliente = new JLabel(Datos.getNombre());

panel.add(tagCliente);

JLabel tagDias = new JLabel(Datos.getDias());

panel\_1.add(tagDias);

JLabel tagModelo = new JLabel(Datos.getModelo() != null ? Datos.getModelo().toString() : "No seleccionado");

panel\_2.add(tagModelo);

JLabel tagJubilado = new JLabel(Datos.isJubilado() ? "Sí" : "No");

panel\_3.add(tagJubilado);

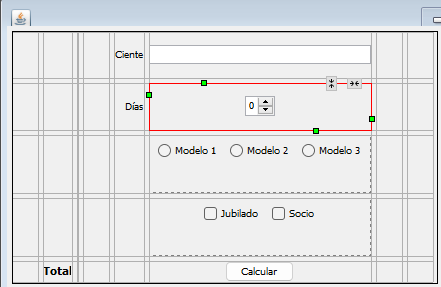
JLabel tagSocio = new JLabel(Datos.isSocio() ? "Sí" : "No");

panel\_4.add(tagSocio);

}

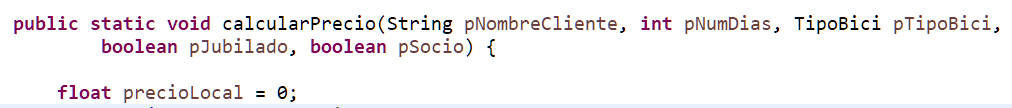
**Ejercicio 3 Ventanas resuelto - Tienda de bicicletas**

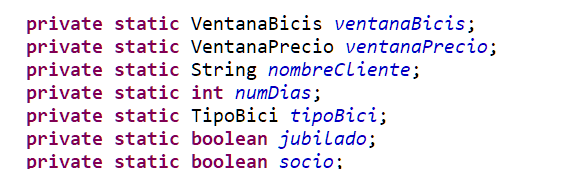
Tras crear la interfaz vamos al Main, llamamos a las ventanas y metemos los datos. Para el número de días hemos usado int, y en la interfaz, un objeto de tipo spinner para los días. Es importante decir que los checkboxes son booleanos y los radio buttons, enumerados.

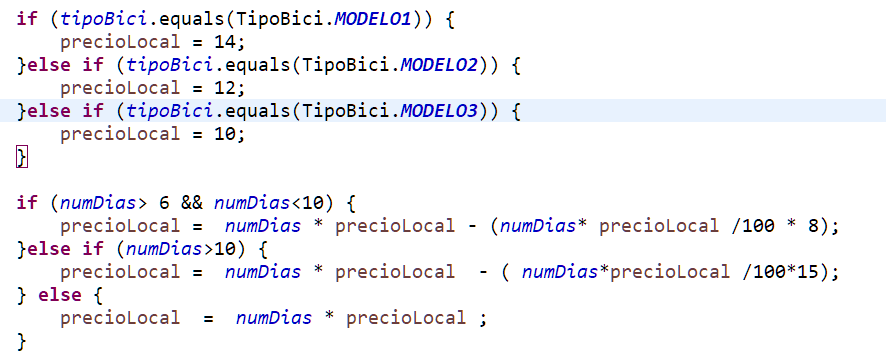


El Main se encargará de tener todos los métodos y la información que se genera a través del modelo. Esto último solo se usa si queremos crear muchos objetos del mismo tipo, por ejemplo usuarios.

Vamos a crear una función en Main, que es la que se encargará de calcular el precio y a la que pasaremos por parámetro todos los datos. En este punto añadiremos a los datos una variable global que guarde el precio, pero no lo pasaremos como parámetro ya que es un parámetro de salida, no de entrada: es lo que se va a calcular.





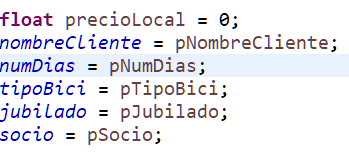


En el código de arriba recogemos primero los enumerados. Si el modelo seleccionado es igual al modelo 1 entonces establecemos un precio, y así sucesivamente. El precio en sí se calcula multiplicando los días por el precioLocal (que es la variable que hemos establecido dentro del método) y a esto restándole el descuento, que es los días \* el precio dividido entre 100 \* el número del porcentaje.

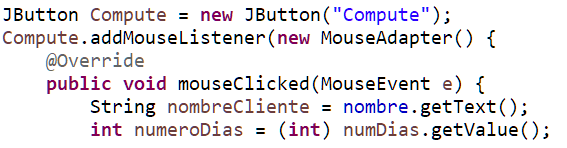
No usamos returns porque estaríamos devolviendo datos a una ventana que vamos a cerrar, así que la función es de tipo void. Y en el caso de usar returns solo hay que usar uno, es mejor poner una variable local antes y devolver finalmente dicha variable local.

Una vez hecho esto solo queda guardar la variable local en la global que hemos definido más arriba. Haremos lo mismo con las otras variables.

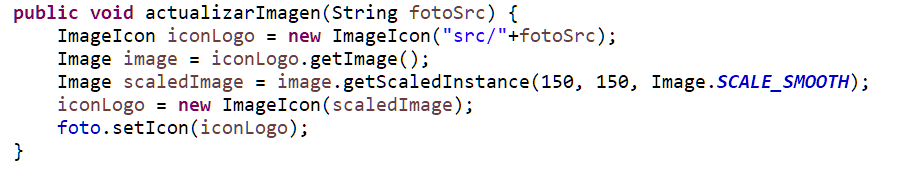


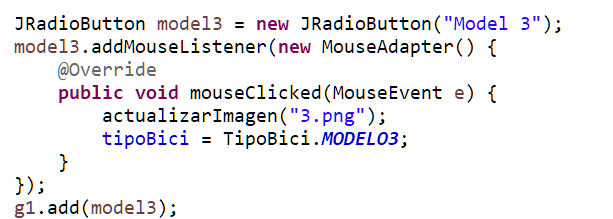


En la ventana que contiene la tienda en sí, en el evento del botón calcular, recogemos el valor del nombre y el número de días y lo guardamos en un variable.

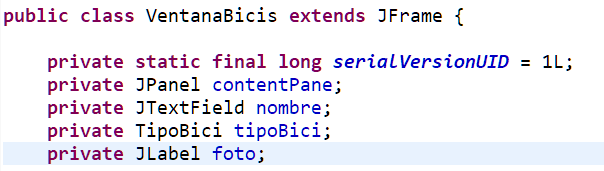


Para los radio buttons podemos añadirlos manualmente o no. Para actualizar la imagen de las bicis, en el evento de cada botón llamaremos al método encargado. Pasaremos por parámetro el nombre del archivo y eso es lo que cambiará en cada método.



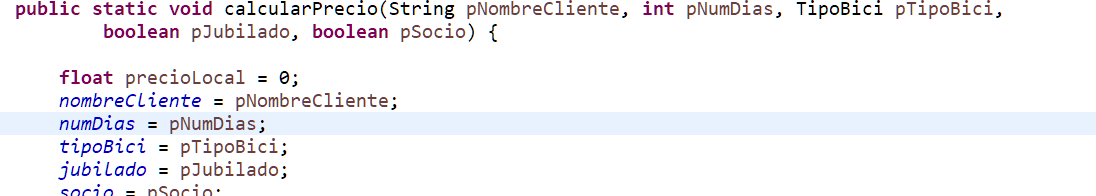


Hay que crear un jLabel en el constructor para la foto ya que necesitamos una variable desde la que poder acceder desde el método.

****

También hay que guardar en una variable global el modelo para posteriormente poder calcular el precio. Usamos para ello una variable de tipo enum, también en el constructor, y en el código donde actualizamos las imágenes guardamos dicho modelo (Ver imagen de arriba).

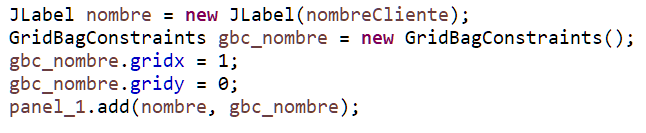
En Main, en el método de calcular, vamos a ir añadiendo por parámetro las variables guardadas. Esto es lo que permitirá pasar datos de una ventana a otra.

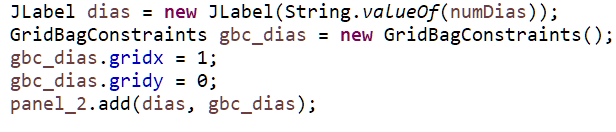


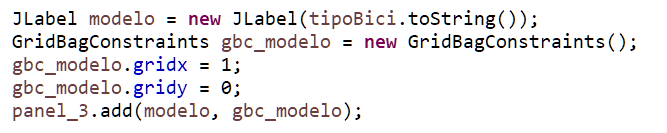
Para pasar datos entre ventanas, en el constructor de la segunda ventana pasamos los parámetros que están ya en Main, en el método calcular..

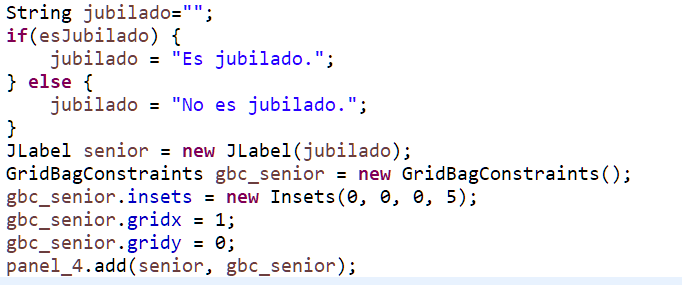


Y en cada JLabel ponemos el parámetro que toque.



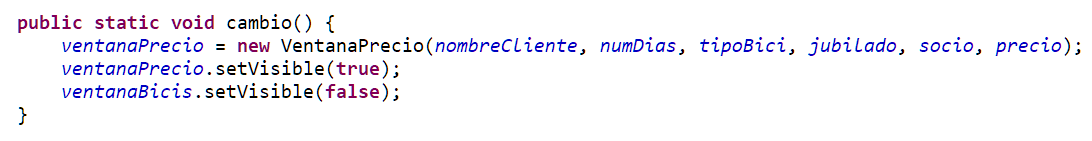






En el caso de los enumerados usamos un método toString, y en el de los booleanos convertimos esto a string, y a la variable global le metemos un condicional que nos muestre, en la ventana, un valor u otro.

Para el precio hay que poner el precio en la lista de parámetros del constructor de la segunda ventana (ver arriba). En la función de cambiar ventana hay que pasar todos los parámetros también, ya que ahora tenemos parámetros.



En el caso de que queramos crear muchos objetos usaremos una clase en Modelo, y en Main tendremos un array de usuarios.

En la función de Calcular de la primera ventana pasamos comoo parámetros los nombres de las labels:

