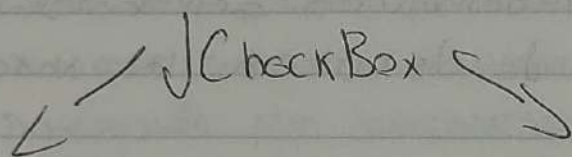


Componentes Swing CHECKBOX

Los CheckBox son las casillas que son de tipo cuadrado y al momento de pulsar en ellas aparecerá un vice y al momento de volver a pulsar el vice se quita.

Esto en las interfaces gráficas en los Formularios es para seleccionar una serie de Items y permite seleccionar varias a la vez. Para construir este tipo de componentes varias a la vez. Para construir este tipo de componentes en Java tenemos que utilizar la clase JCheckBox al cual es muy sencillo y esta clase que consta con una serie de métodos, como por ejemplo el método:



`isSelected()`

`setSelected (Boolean)`

Estos dos métodos no son directamente de la clase JCheckBox, sino que los hereda de la clase abstracta. Estos métodos nos permiten en el caso de '`isSelected()`' determinar si la casilla de verificación está seleccionada o no está y el método '`setSelected`' permite desde el código seleccionar esa casilla o no seleccionarla o quitar la selección.

Esta clase tiene una herencia bastante larga y ha una clase con la siguiente cadena que toma los dos métodos mencionados.

`javax.swing.AbstractButton`.

Componentes Swing Botones De Radio

En este video veremos los botones de radio, estos son comunes en Formularios, en interfaces graficas, porque son como los checkboxes pero redondos. Entonces cuando uno pulsa sobre estos elementos, en vez de aparecer un mouse, va aparecer un redondo negro en el centro indicando que ese elemento esta seleccionado.

Las diferencias que hay en el checkbox y los botones de Radio o conocidos como JRadioButton que es la clase a utilizar para construir este tipo de objetos.

Al igual que ocurre en Formularios web o en otras interfaces graficas que tienen que ver con Java, la diferencia fundamental esta en que los cuadros pueden seleccionar varias a la vez. Entonces tu cuando a un usuario le quieres dar a elegir entre varias opciones a la vez. Es decir que se puedan seleccionar varias opciones como por Ejemplo.

El tipico formulario es el que hay a ficciones y debemos seleccionar varias aficiones lo utilizamos

Pero cuando te das a elegir una unica opcion entre varias y solamente es posible una unica opcion entonces tendras que utilizar los redondos, los botones de radio, como por ejemplo:

El

Un Formulario que te pregunta si genero!

a2223330168

24/01/2024

Componentes Swing Botones De Radio II

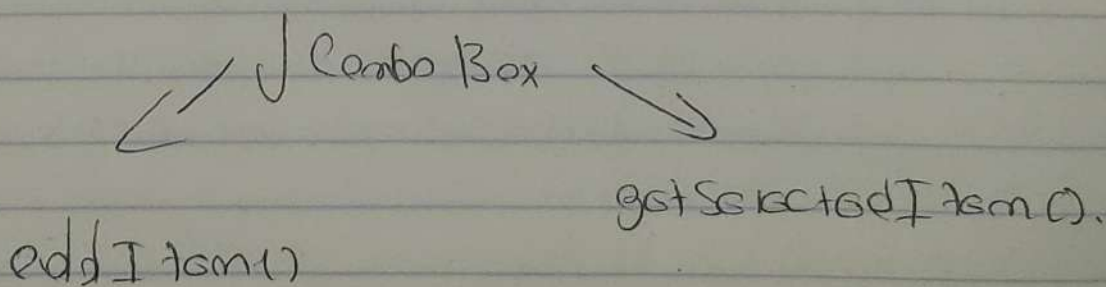
En este video lo que se va es como ademas de agregar los botones de radio les vamos a dar interactividad para que respondan a eventos para que al usuario cuando pulse en estos elementos ocurra algo.

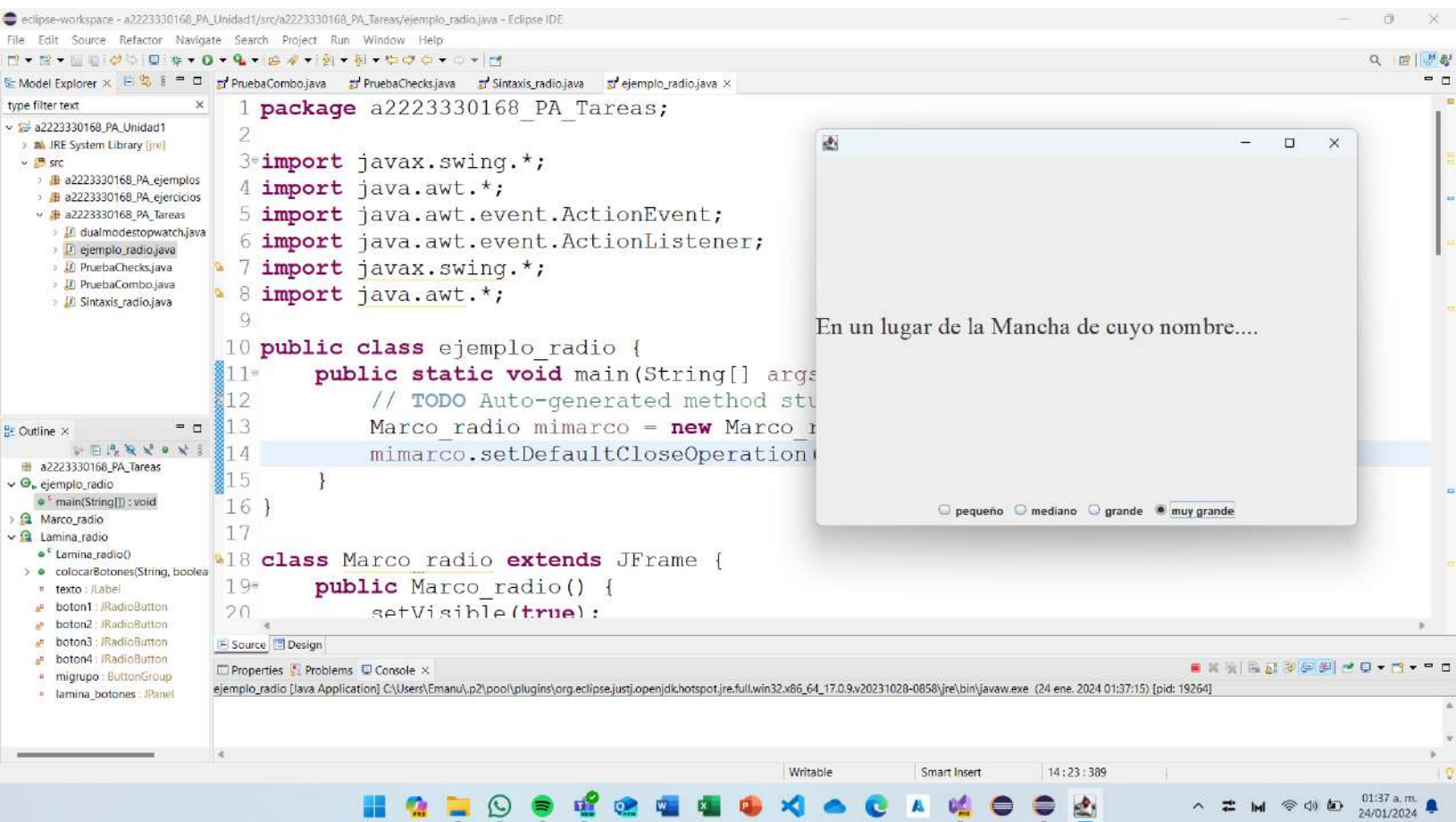
Ademas de elaborar de las formas sencillas en la cual se entenderan facilmente con lo visto en el video anterior y otra manera mas compleja de ver para avanzar, nos ahorra unas cuantas lineas de codigo.

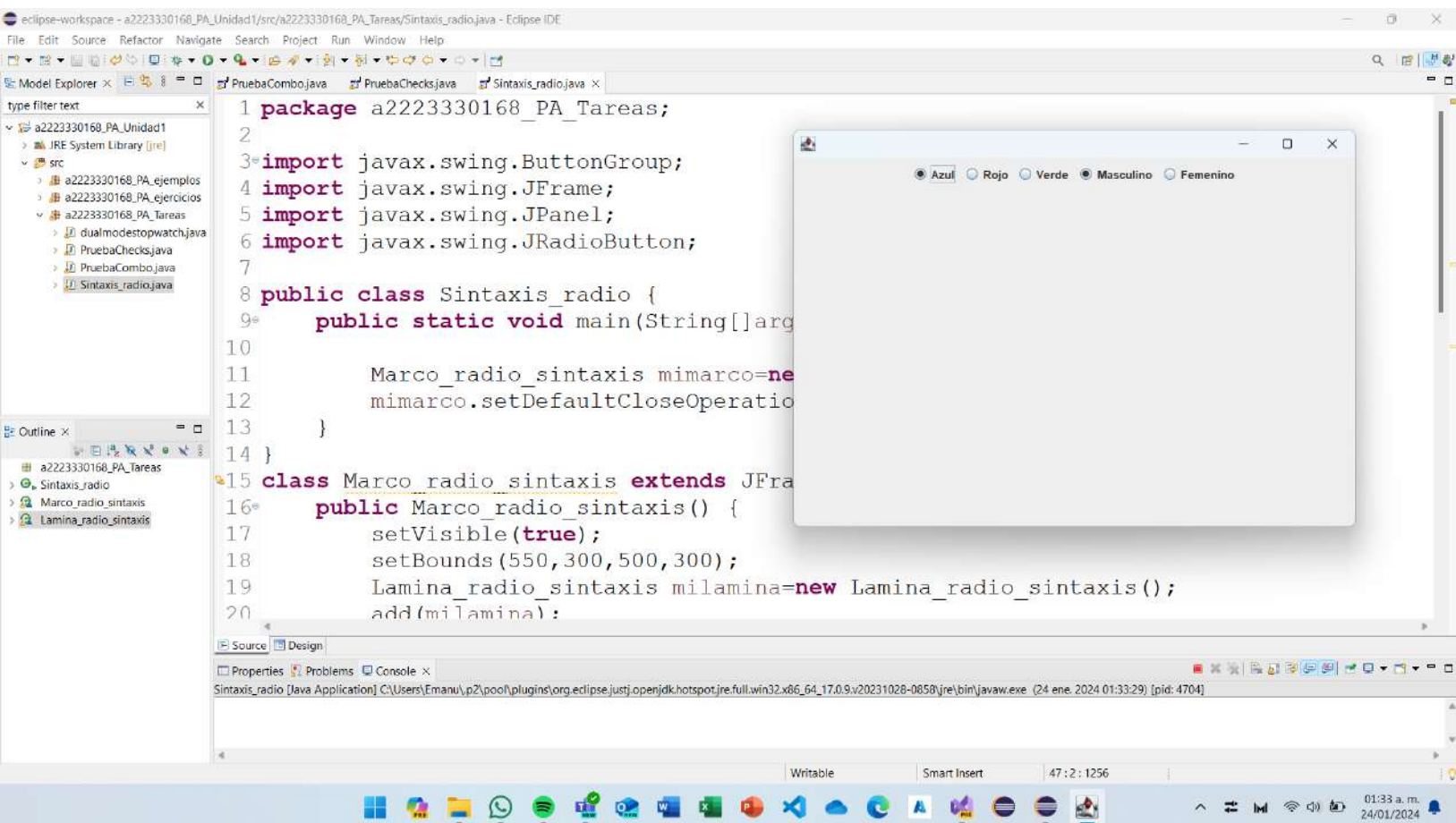
Componentes Swing Combo Box.

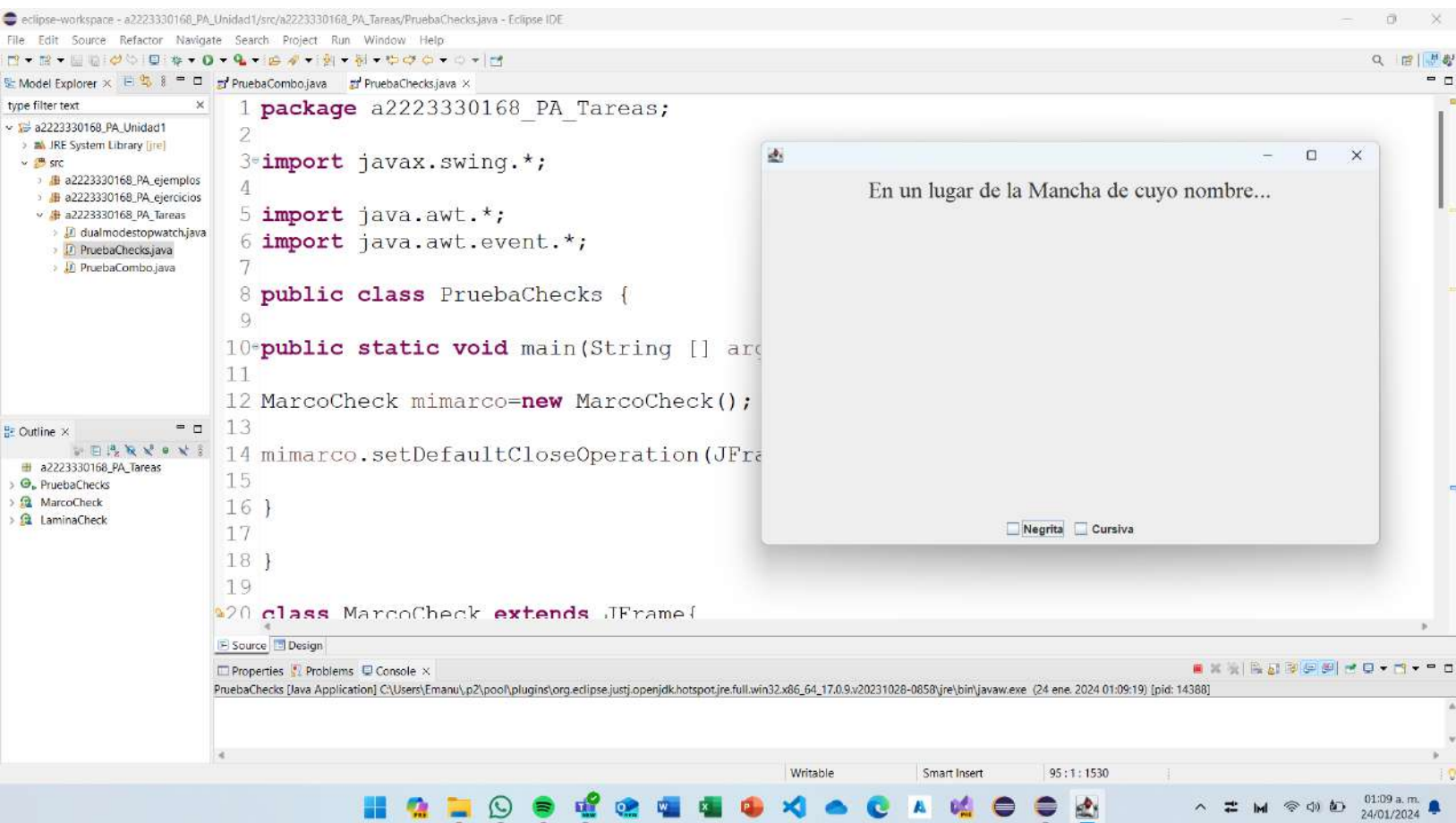
En este video veremos los comboBox los cuales son los tipicos menus desplegables que toda interfaz grafica pues se suele tener para elegir una serie de opciones tambien podemos construir este tipo de elementos con Java.

La clase que utiliza para construir este tipo de componentes es la clase `JComboBox` como todas las clases tienen una serie de metodos, como por ejemplo









eclipse-workspace - a2223330168_PA_Unidad1/src/a2223330168_PA_Tareas/PruebaCombo.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Model Explorer x PruebaCombo.java x PruebaChecks.java

type filter text

a2223330168_PA_Unidad1

- JRE System Library [jre]
- src
 - a2223330168_PA_ejemplos
 - a2223330168_PA_ejercicios
 - a2223330168_PA_Tareas
 - dualmodestopwatch.java
 - PruebaChecks.java
 - PruebaCombo.java

Outline x

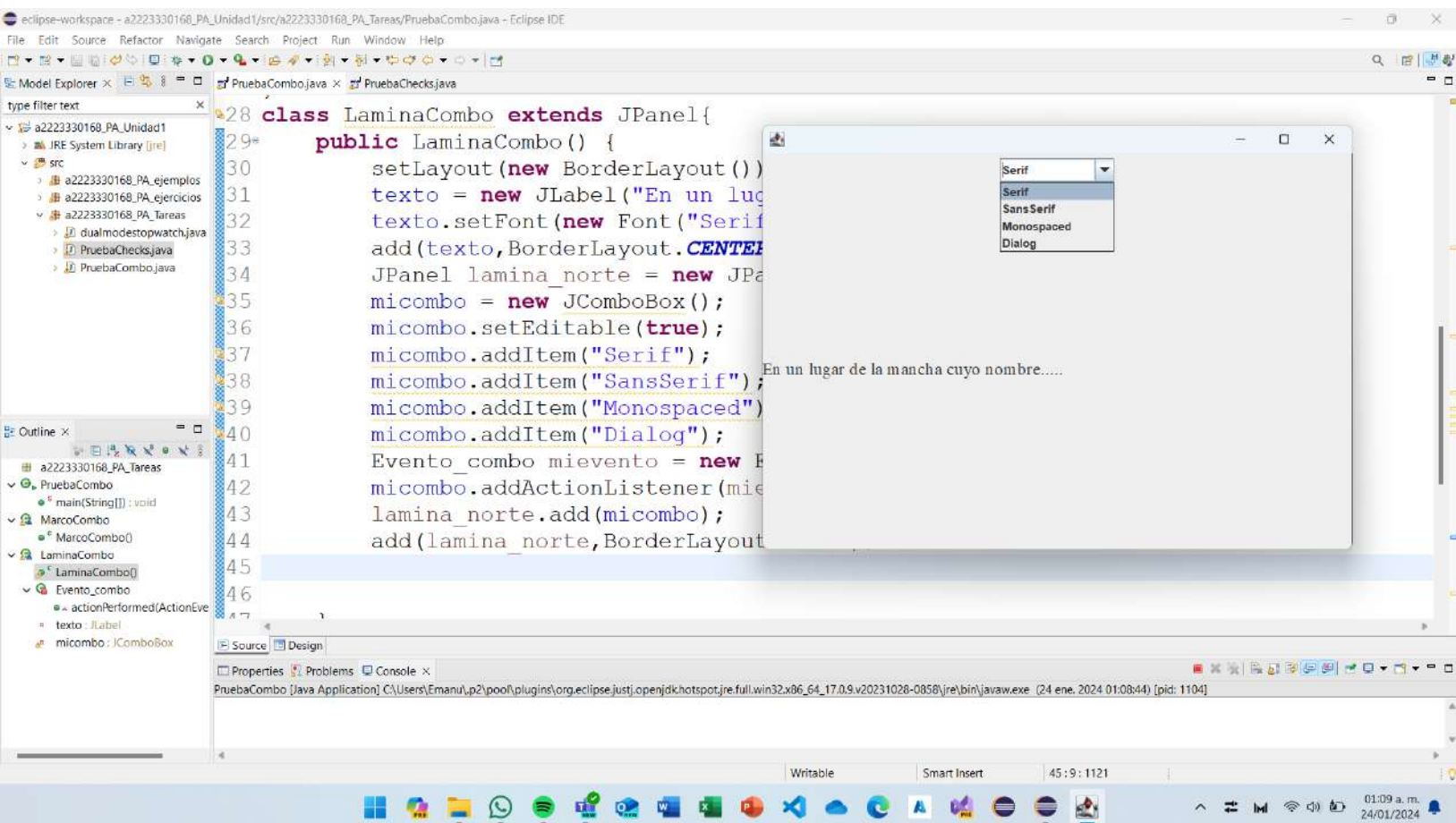
- a2223330168_PA_Tareas
 - PruebaCombo
 - main(String[]): void
 - MarcoCombo
 - MarcoCombo()
 - LaminaCombo
 - LaminaCombo()
 - Evento_combo
 - actionPerformed(ActionEvent)
 - texto: JLabel
 - micombo: JComboBox

```
28 class LaminaCombo extends JPanel{
29     public LaminaCombo() {
30         setLayout(new BorderLayout());
31         texto = new JLabel("En un lugar de la mancha cuyo nombre....");
32         texto.setFont(new Font("Serif", Font.PLAIN, 16));
33         add(texto, BorderLayout.CENTER);
34         JPanel lamina_norte = new JPanel();
35         micombo = new JComboBox();
36         micombo.setEditable(true);
37         micombo.addItem("Serif");
38         micombo.addItem("SansSerif");
39         micombo.addItem("Monospaced");
40         micombo.addItem("Dialog");
41         Evento_combo mievento = new Evento_combo();
42         micombo.addActionListener(mievento);
43         lamina_norte.add(micombo);
44         add(lamina_norte, BorderLayout.NORTH);
45     }
46 }
```

PruebaCombo [Java Application] C:\Users\Emanuelp2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (24 ene. 2024 01:08:44) [pid: 1104]

Writable Smart Insert 45:9:1121

01:09 a.m. 24/01/2024



The screenshot displays the Eclipse IDE interface. The main editor shows the Java code for the `LaminaCombo` class, which extends `JPanel`. The code initializes a `JLabel` with the text "En un lugar de la mancha cuyo nombre....", sets its font to "Serif", and adds it to the panel. It also creates a `JComboBox` with items "Serif", "SansSerif", "Monospaced", and "Dialog", sets it to be editable, and adds it to the panel. A preview window titled "PruebaCombo" is overlaid on the code, showing the graphical user interface. The GUI consists of a light gray panel with a white label containing the text "En un lugar de la mancha cuyo nombre...." and a white combobox below it. The combobox currently displays "Serif" and has a dropdown arrow. The IDE's left sidebar shows the Project Explorer and Outline views, and the bottom status bar indicates the current file, line, and column.