Módulo 4. JFrame (visual) en Framework

CÓRDOBA CREA FUTURO

Introducción

Introducción

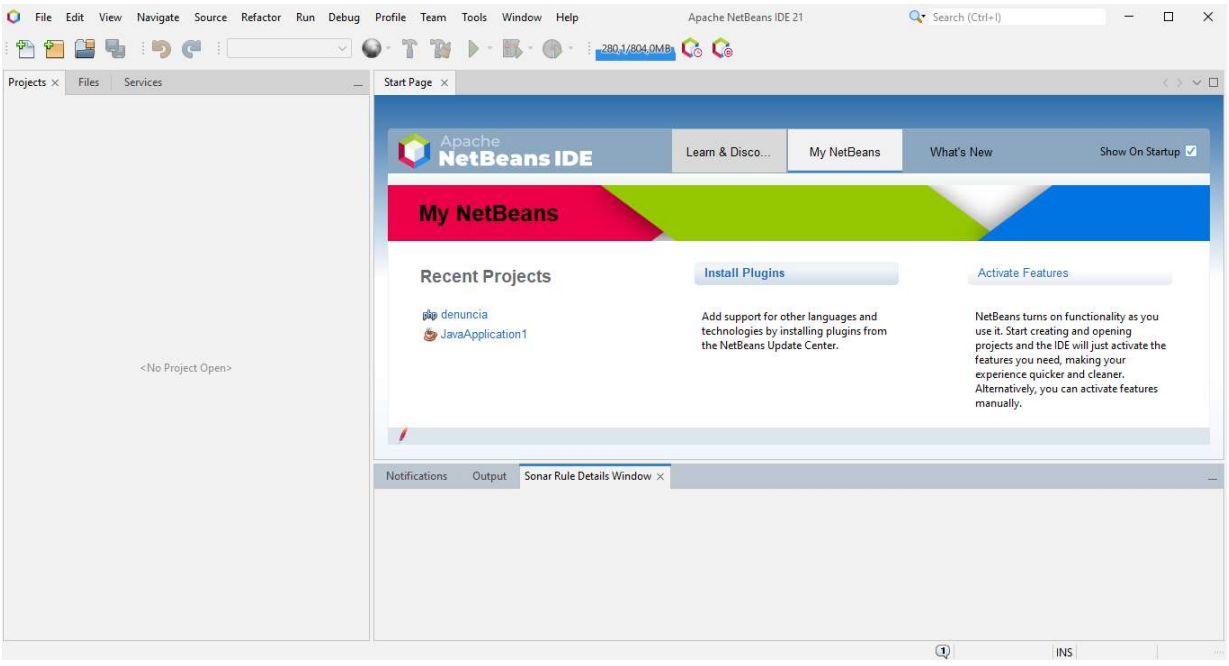
Un **JFrame** en Java es la ventana principal de una aplicación gráfica (GUI: interfaz gráfica de usuario) dentro del *Framework* Java *Swing*, sirviendo como un contenedor de nivel superior (marco) donde se añaden otros componentes visuales como botones, etiquetas y paneles para crear la interfaz, manejando eventos y propiedades como tamaño, posición y comportamiento al cerrarse. Es fundamental para construir aplicaciones de escritorio visuales, permitiendo un diseño que pueda arrastrar y soltar en IDEs como NetBeans y una personalización avanzada de su aspecto y comportamiento.

En Java, JPanel es un componente fundamental de la interfaz gráfica de usuario (GUI) de *Swing*, actuando como un contenedor ligero para agrupar y organizar otros elementos (botones, etiquetas, etc.) dentro de una ventana (JFrame), permitiendo un mejor diseño y distribución de los

elementos visuales para controlar su movimiento y posicionamiento de forma lógica, mejorando la estructura de la UI (interfaz de usuario).

Un **JFrame** es la base visual para implementar patrones de diseño como el **modelo-vista-controlador (MVC)** en aplicaciones Java, sirviendo como la «vista» que el usuario manipula, mientras que el código detrás del JFrame maneja la lógica (el «controlador»).

Figura 1: IDE NetBeans



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

CONTINUAR

1. JFrame

1. JFrame

Los puntos sobresalientes de JFrame son los que mencionaremos a continuación.

- **Ventana principal**: es el contenedor más alto en una jerarquía de *Swing*, similar a una ventana del sistema operativo, como por ejemplo Windows o macOS (con barra de título, minimizar, etc.).

- **Contenedor de contenido**: tiene un «panel de contenido» (getContentPane) donde se colocan otros elementos, permitiendo organizar la interfaz y donde se colocan otros elementos (JComponents) como JButton, JLabel, JPanel, etc.

- **Componentes visuales**: permite agregar componentes (JLabels, JButtons) y

contenedores (JPanel) para construir la UI (*user interface* o interfaz de usuario).

- **Diseño visual**: se puede diseñar visualmente en un IDE, arrastrando y soltando componentes, y el código generado configura sus propiedades.

- **Manejo de eventos**: controla acciones del usuario, como cerrar la ventana (setDefaultCloseOperation) o interactuar con elementos internos.

- **Personalización**: se puede configurar su tamaño (setBounds), visibilidad (setVisible(true)), icono, imagen de fondo y comportamiento.

- **Parte de *Swing***: pertenece a la biblioteca javax.swing, el conjunto de componentes de interfaz gráfica de usuario (GUI) de Java.

Características clave y su uso visual

- **Diseño visual (*drag-and-drop*):** en entornos como NetBeans, puedes arrastrar y soltar

componentes directamente en un JFrame para diseñar visualmente tu interfaz.

- **Panel de contenido (*content pane*):** el JFrame tiene un panel interno (el *content pane*) donde realmente se añaden los componentes; esto es clave para organizar la UI.

- **Manejo de eventos**: define cómo responde la ventana a acciones del usuario y cómo cerrar (usando setDefaultCloseOperation).

- **Personalización**: puedes extender la clase JFrame para crear formularios complejos, cambiar su tamaño, añadir imágenes de fondo, menús (JMenuBar), o incluso hacerlo transparente y arrastrable.

Relación con el *framework*

- **Java *Swing***: JFrame pertenece a *Swing*, la librería estándar de Java para interfaces gráficas (GUI), parte del JDK.

- **IDE (NetBeans/eclipse):** los Entornos de Desarrollo Integrado facilitan la creación de JFrames mediante herramientas gráficas, generando el código base.

- **Contenedores (JPanel):** a menudo, se usa un JPanel dentro del JFrame para organizar componentes o aplicar una imagen de fondo, ofreciendo más flexibilidad.

En resumen, JFrame es el esqueleto de la ventana en aplicaciones de escritorio Java, permitiendo la creación de interfaces ricas y personalizadas gracias a los componentes y herramientas del *Framework*.

¿Qué es UI y cómo encaja JPanel?

- **UI (*User interface*/Interfaz de usuario):** es el espacio de interacción entre el humano y la máquina, usando elementos gráficos como ventanas, botones y menús para que el usuario controle la aplicación.

- **JPanel (panel de *Swing*):** es una pieza clave para construir la UI en Java.

- **Contenedor**: no es una ventana visible por sí misma, sino una «caja» invisible dentro de un JFrame (la ventana principal).
- **Organización**: se usa para agrupar componentes relacionados (por ejemplo, un conjunto de campos de un formulario) y aplicarles un diseño (*Layout Manager*) específico, facilitando su posicionamiento y movimiento.
- **Estructura**: permite dividir la interfaz en secciones manejables, mejorando la organización y la experiencia de usuario.

Es decir, un JPanel es como una subsección o un «mini lienzo» dentro de tu ventana principal (JFrame), donde se pueden colocar otros elementos de la interfaz (como JButton, JLabel, JTextField) y organizarlos con Layout Managers para que se vean y se comporten como deseamos.

Usando el IDE NetBeans, se mostrarán los pasos para armar un formulario JFrame donde, en primer lugar, es necesario crear un proyecto mediante Java Application. El formulario JFrame dependerá de Java Application.

CONTINUAR

2. Desarrollando nuestra aplicación visual

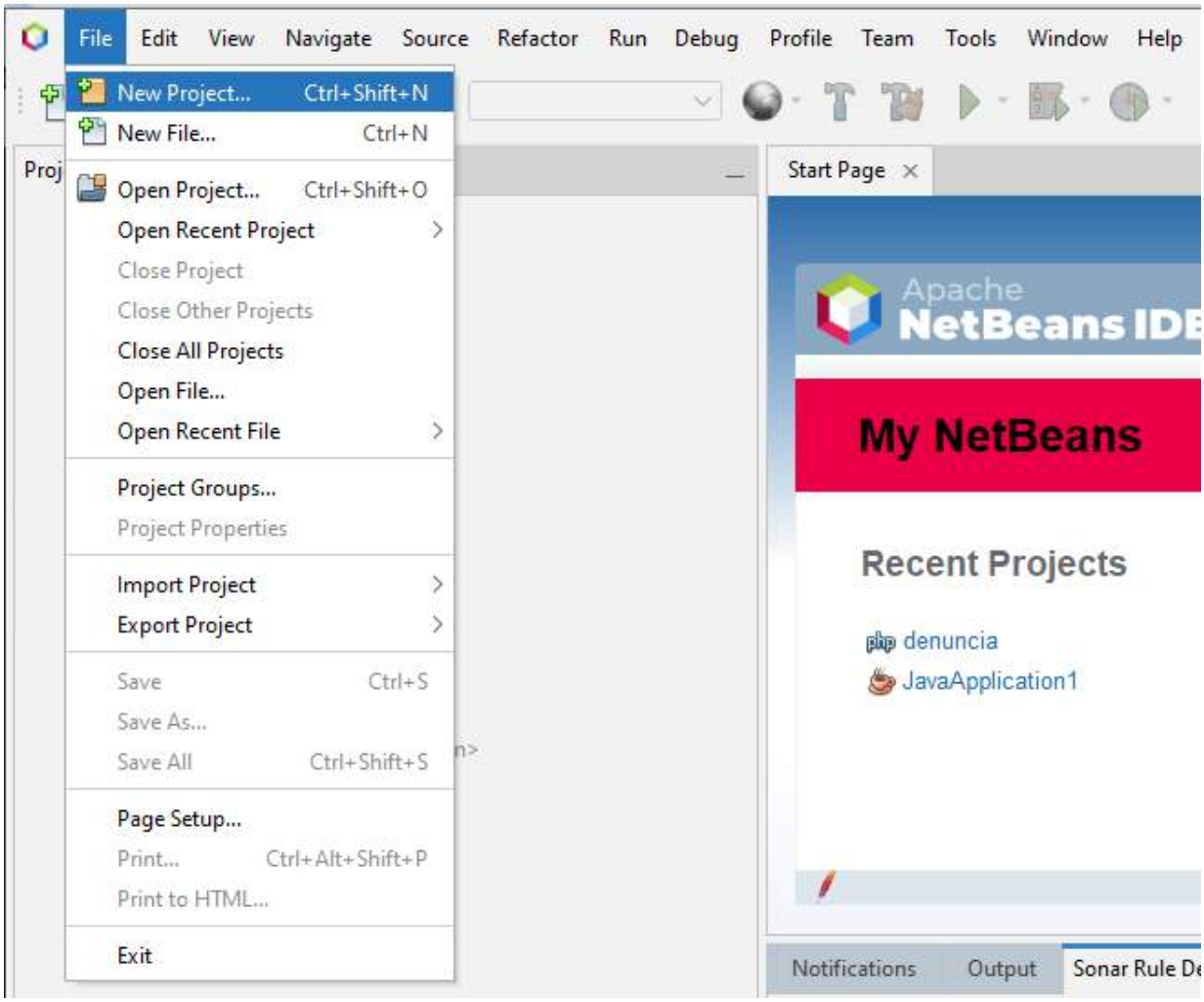
2. Desarrollando nuestra aplicación visual

Paso 1: ejecutar IDE Netbeans, como vemos en la figura 1. Si no lo han descargado, háganlo desde el siguiente enlace:

https://netbeans.apache.org/front/main/download/

Paso 2: para crear un nuevo proyecto, seleccionamos ***File*** y luego ***New Project***, como veremos en la siguiente figura.

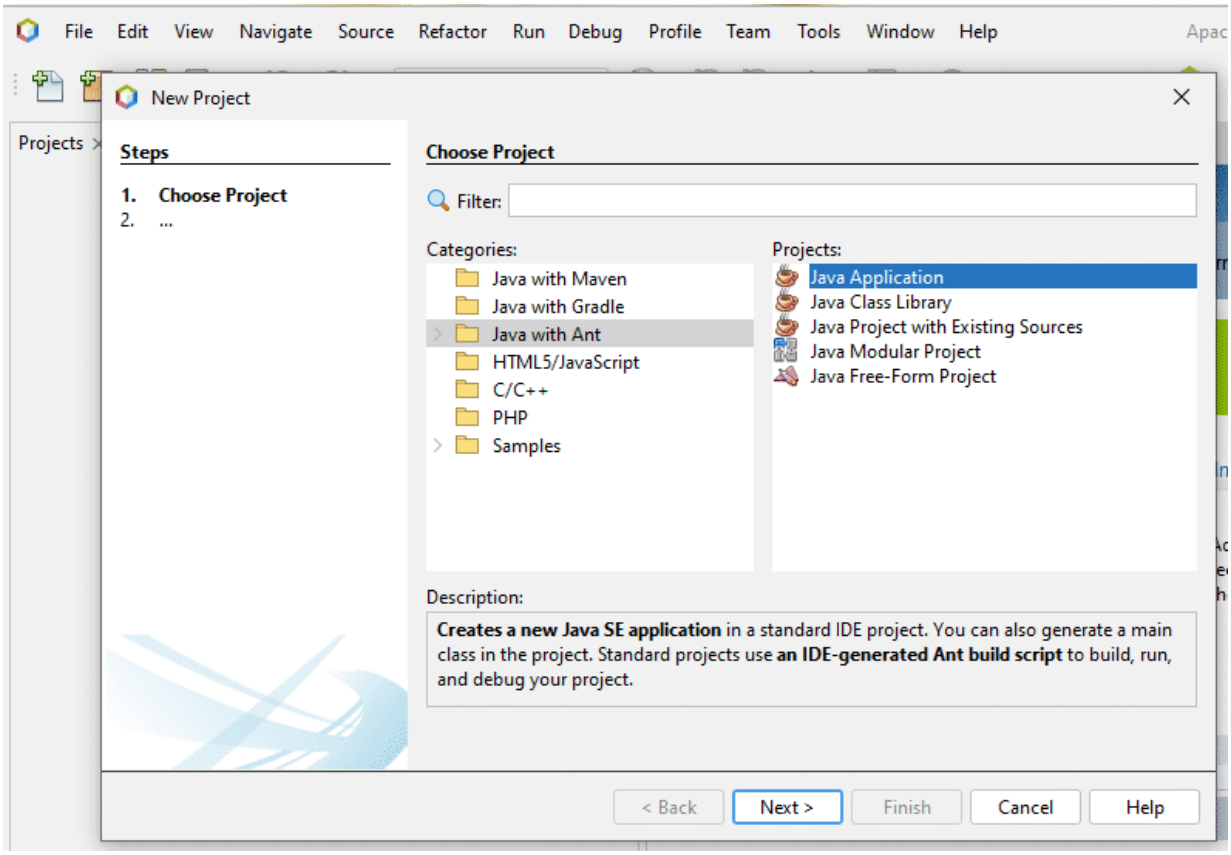
Figura 2: IDE Netbeans 21, creamos un nuevo proyecto



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Paso 3: navegamos para crear una **Java Application** y luego seleccionamos ***Next***, como veremos en la siguiente figura.

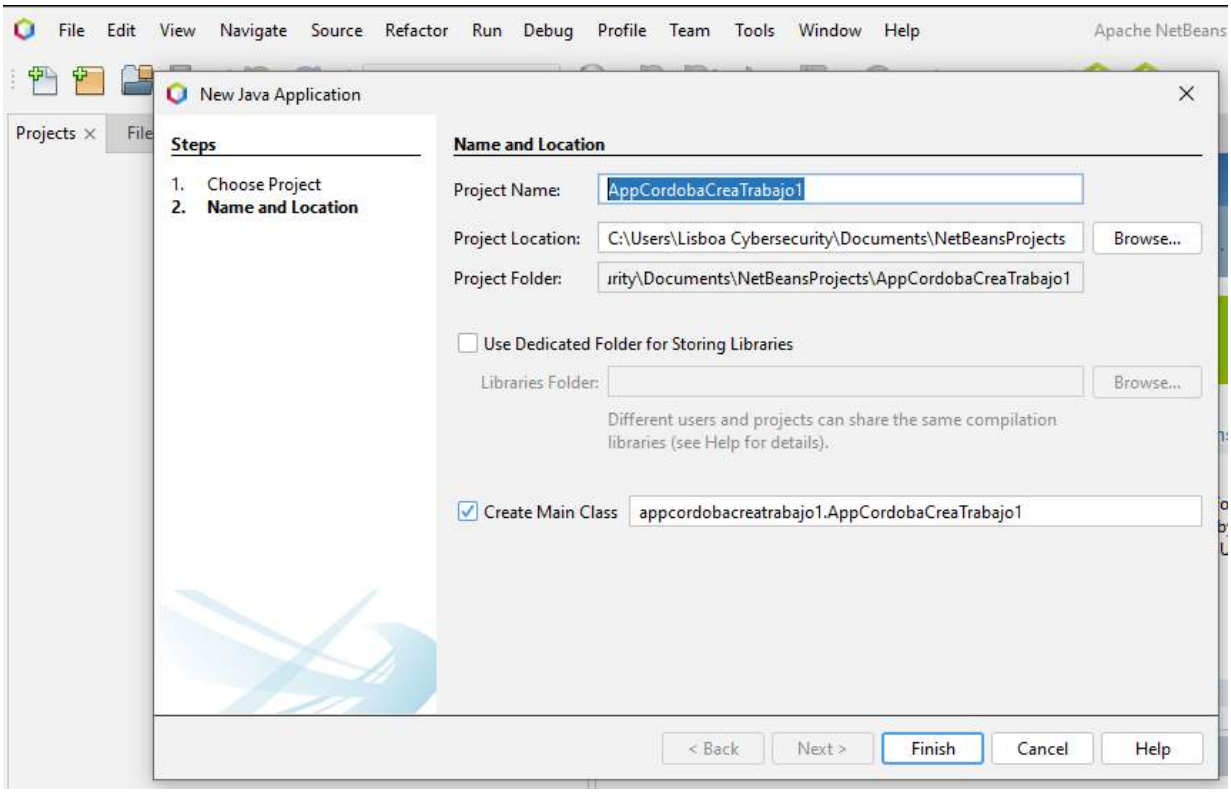
Figura 3: Creando una aplicación nueva



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Paso 4: en ***Project Name***, le ponemos un nombre a la aplicación; en este ejemplo le pondremos de nombre **AppCordobaCreaTrabajo1,** y luego presionamos el botón ***Finish***. Por el momento, los demás parámetros no serán tocados, dejándolos tal como están.

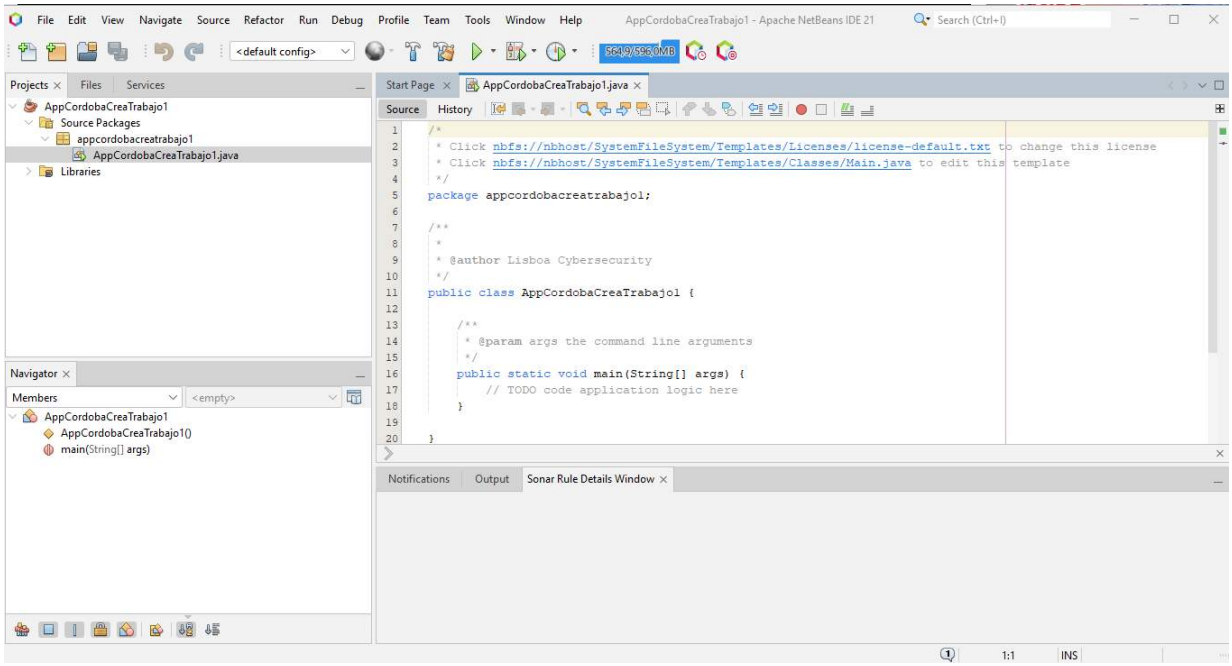
Figura 4: Poniendo nombre a nuestra aplicación



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Seguidamente, visualizarás algo similar a lo que muestra la figura 4.1 a continuación.

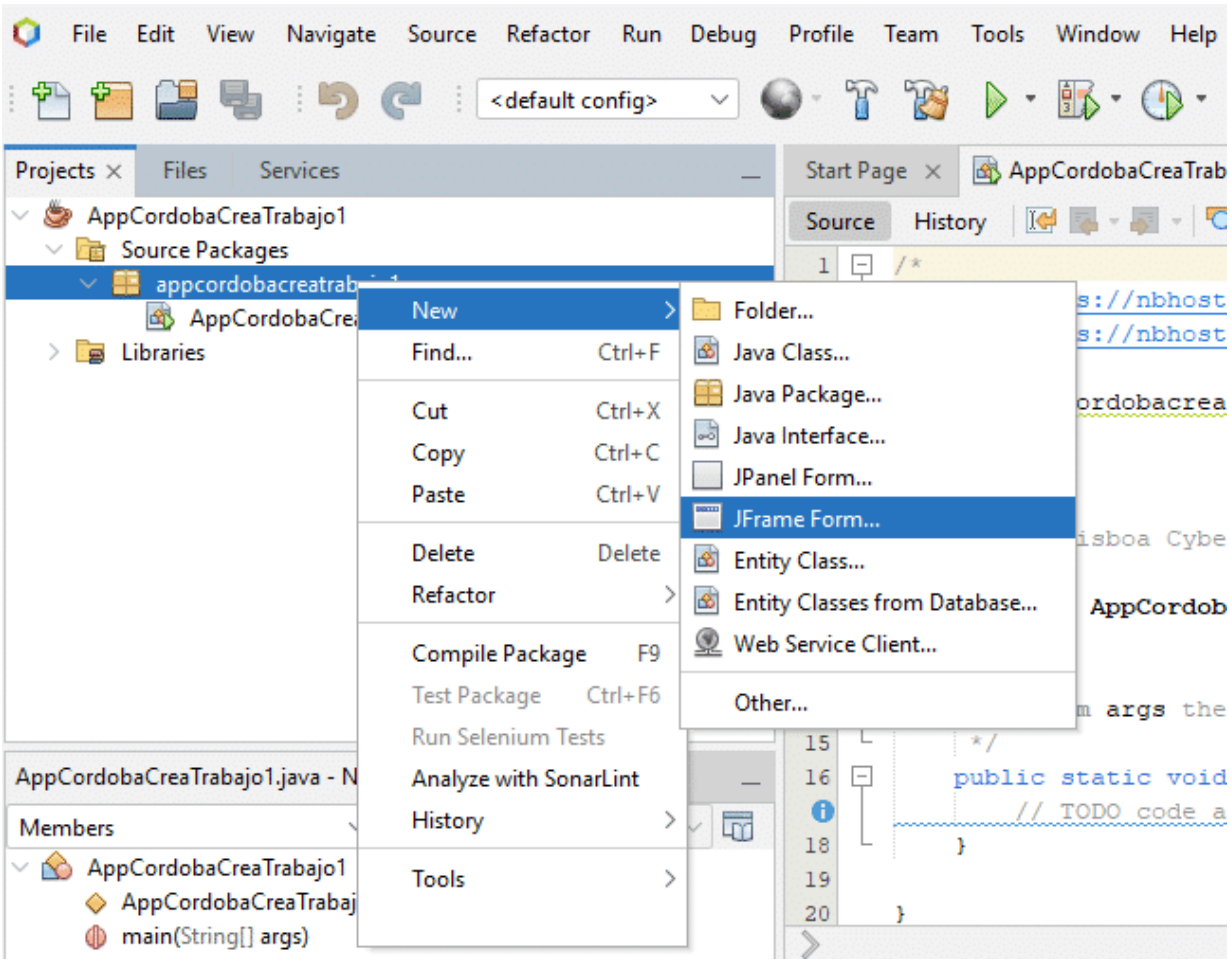
Imagen 4.1: La aplicación creada



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Paso 5: crearemos el formulario presionando el botón derecho en **AppCordobaCreaTrabajo1**, luego seleccionamos ***New***, navegamos hasta **JFrame** ***Form*** y presionamos el botón izquierdo.

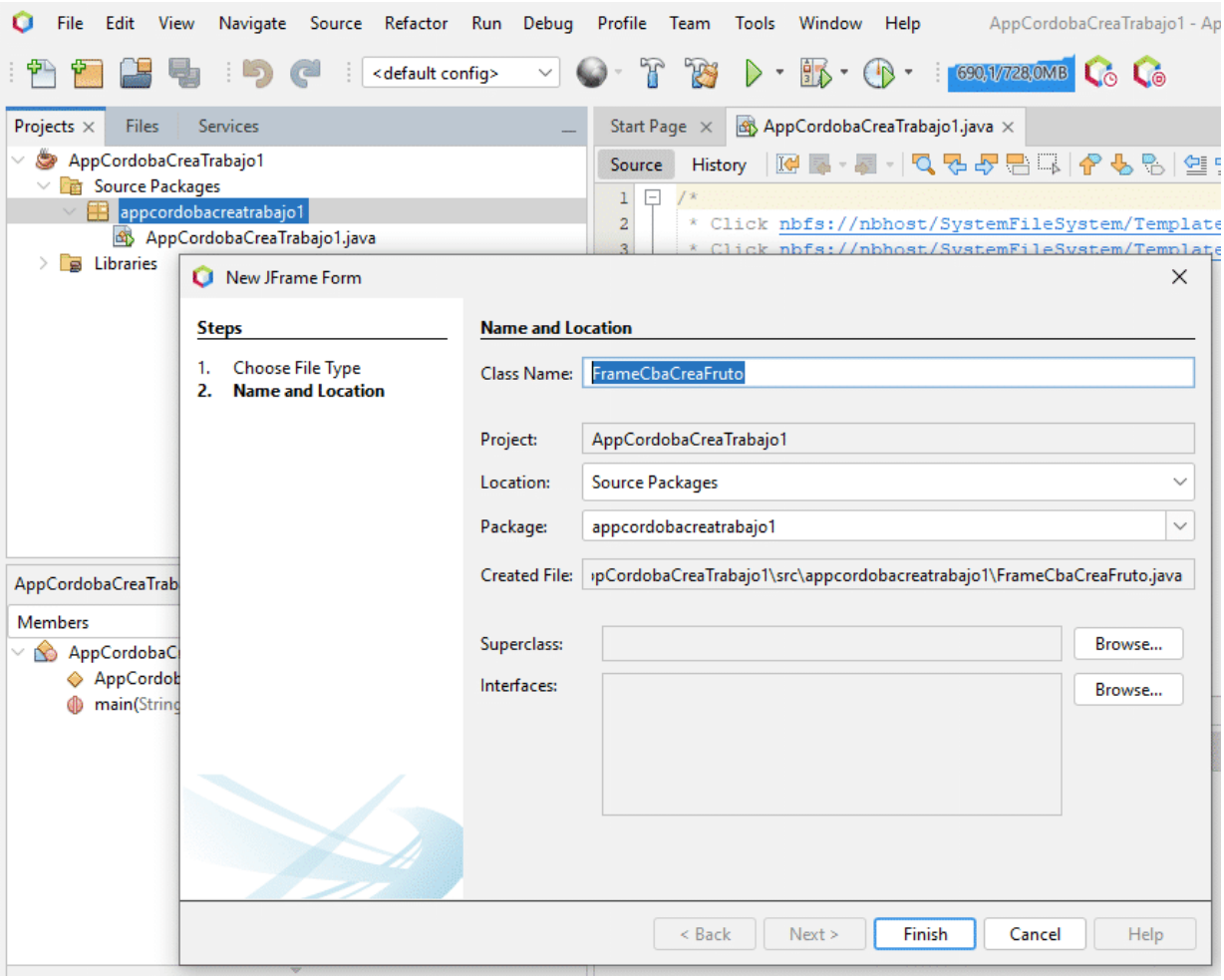
Figura 5: Seleccionar crear JFrame ***Form***



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Paso 6: le pondremos un nombre al *Frame*, en nuestro caso, en *Class Name* escribimos **FrameCbaCreaFruto** y luego le damos ***Finish***. Por el momento, dejaremos al resto de los parámetros tal y como están, sin cambiar nada.

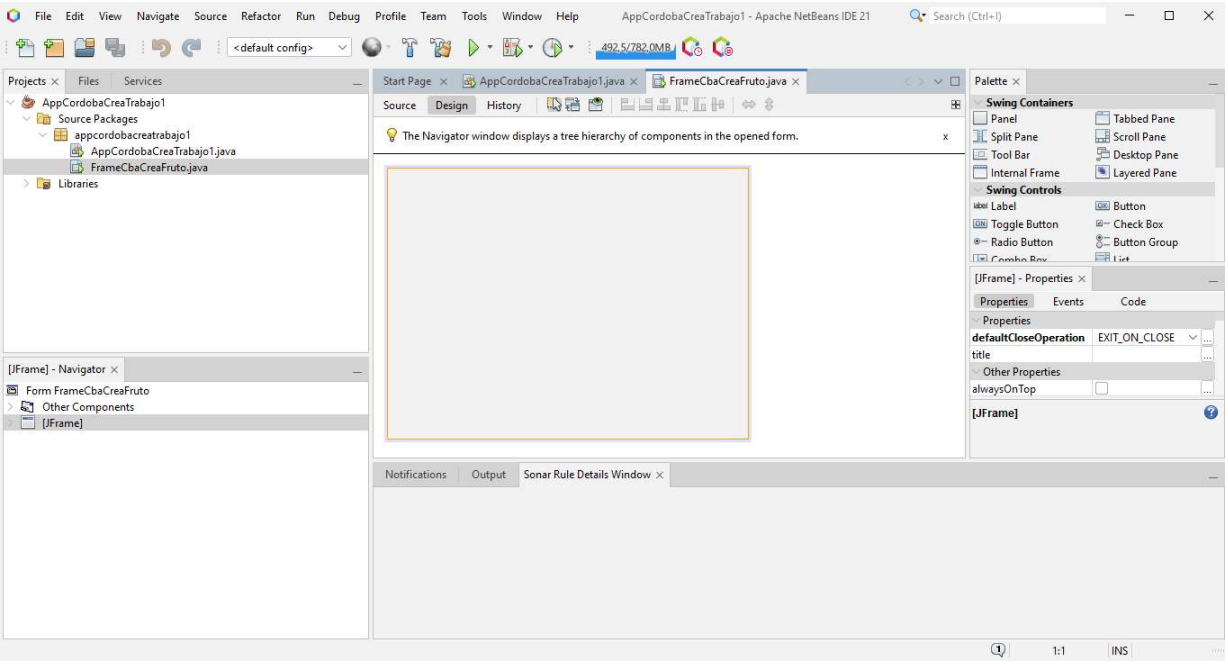
Figura 6: Ponemos nombre al JFrame



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

En la figura 6.1, a continuación, se visualiza el resultado de la creación del JFrame.

Figura 6.1: JFrame creado con el tamaño que viene por defecto



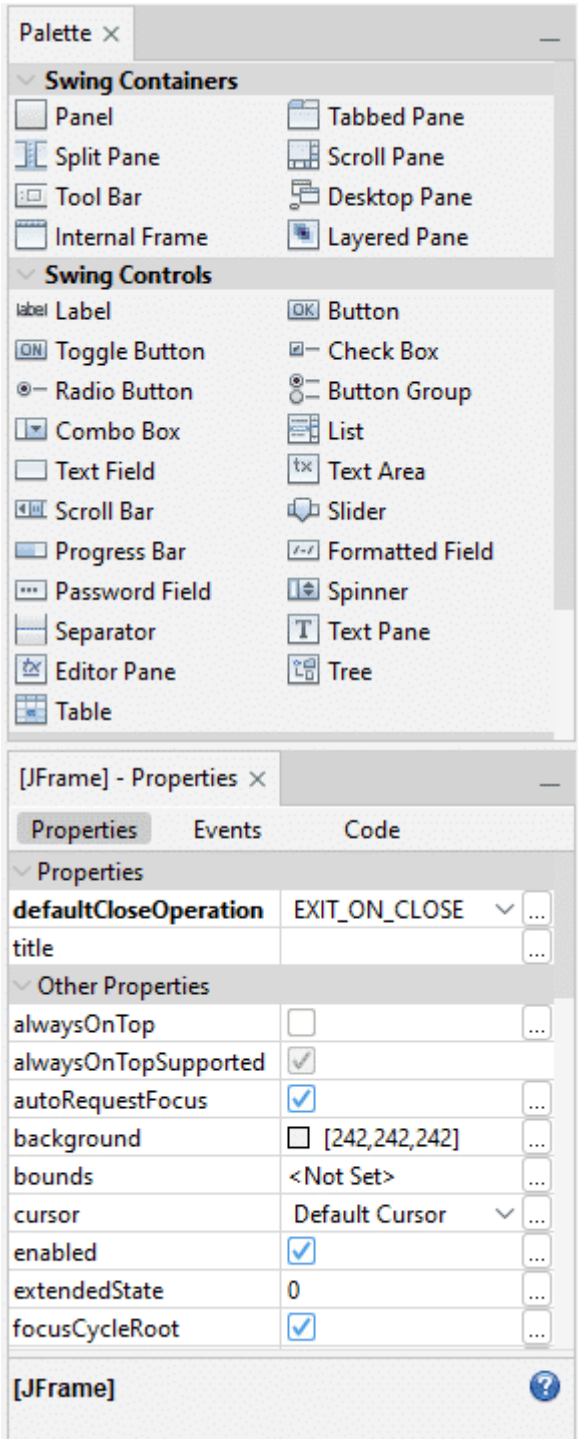
Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

En la figura 6.1 se puede observar que, por defecto, se muestra ***Design*** y la ***Palette*** que se explicarán seguidamente.

Palette

La paleta (*Palette*) en NetBeans es una ventana de componentes visuales (botones, etiquetas, campos de texto, etc.) que se usa en el diseñador de interfaces gráficas (GUI) para arrastrar y soltar elementos en tus formularios (como JFrame o JPanel), organizados por categorías para crear interfaces de usuario de forma rápida y visual, sin necesidad de escribir todo el código manualmente.

Figura 7: *Palette*



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

¿Para qué sirve?

- **Arrastrar y soltar**: permite seleccionar un componente visual (como un JButton o JLabel) y colocarlo directamente en tu diseño.
- **Organización**: los componentes están agrupados en categorías (*Swing*, AWT, contenedores) para facilitar su búsqueda.
- **Diseño visual**: ayuda a construir la interfaz de tu aplicación de escritorio de manera intuitiva, como si fuera un dibujo.

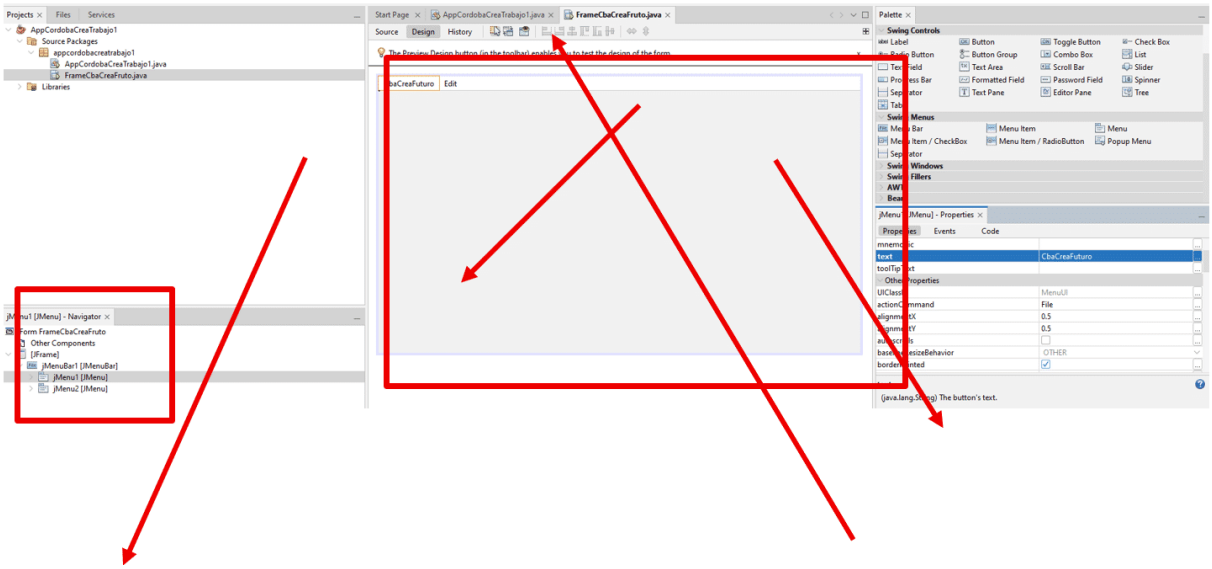
Es una caja de herramientas para construir interfaces gráficas en NetBeans, ofreciendo una forma visual y eficiente de añadir elementos a tus aplicaciones Java.

Ya estamos listos para que la aplicación que vamos a desarrollar tenga botones, menú de opción y todo lo que la

creatividad nos permita traer a la mente que pueda hacer cumplir con el objetivo que se desea satisfacer.

Paso 7: agrandar el marco principal y el *Palette* (figura 8); se puede observar que contiene ***Swing Containers***, ***Swing Controls***, ***Swing Menus***, etc., y estos, a su vez, ofrecen distintas opciones, como por ejemplo ***Swing Menus***: *Menu Bar, Menu Item, Menu*, etc. Arrastrar al marco principal *Menu Bar* de ***Swing Menus*** (por defecto tiene dos opciones: *file* y *edit*). Luego, como se observa en la figura 8, seleccionar la opción jMenu1 para luego, en el *Palette*, ***Properties,*** cambiar el texto a «CbaCreaFuturo».

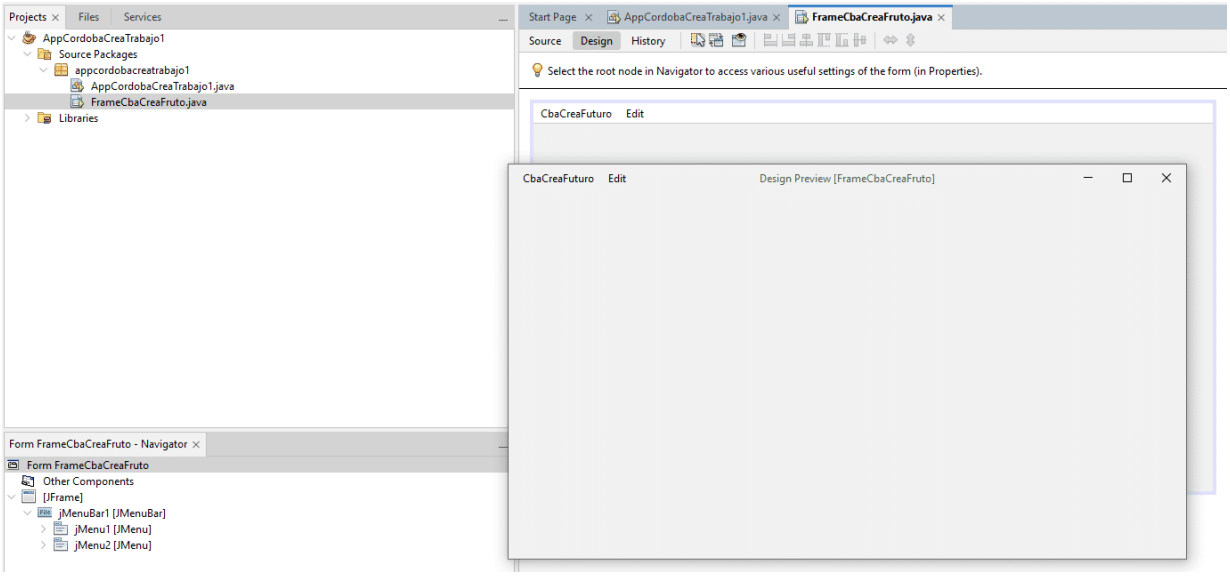
Figura 8: Se agrega menú y se cambia el texto



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Si deseamos ver cómo va quedando la aplicación, podemos hacerlo en el ***Preview*** presionando el botón con el ojo. El resultado se ve como lo que muestra a continuación la figura 8.1.

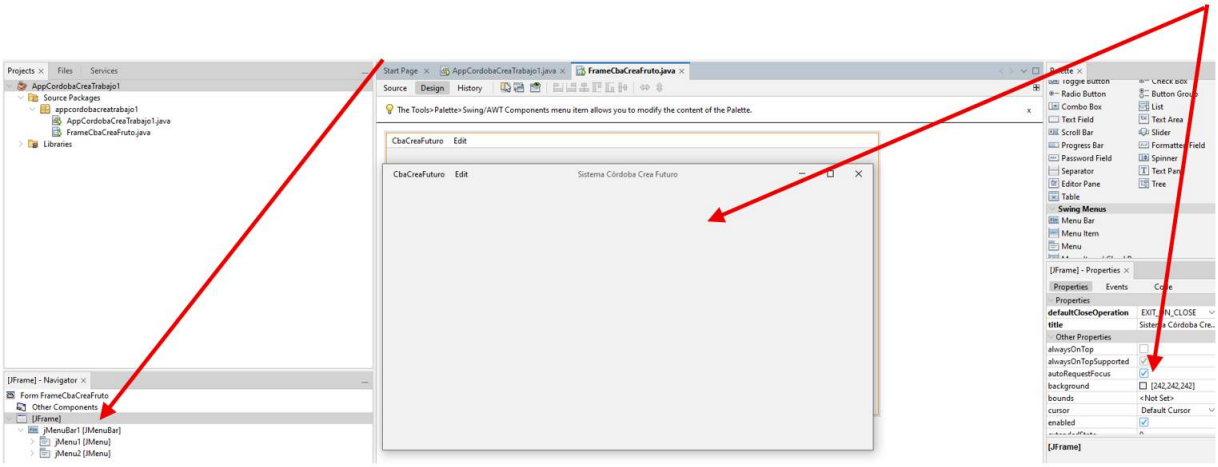
Figura 8.1: *Desing Preview* **de la aplicación llamada [FrameCbaCreaFuturo]**



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Si deseamos cambiar el título de la aplicación para que esté personalizado, puede hacerse en ***title,*** marcando en primer lugar **JFrame** en el ***Navigator.***

Figura 8.2: *Desing Preview* **cambiando el título de la aplicación**



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Paso 8: hasta ahora hemos demostrado de qué manera se puede ir armando el diseño del sistema que estamos simulando, donde se han agregado elementos tales como menú, botones (ver figura 9), etc., con el simple arrastre desde el *Palette*, y, a su vez, también se mostró que por cada elemento que se agrega tenemos las siguientes partes.

- ***Properties*** (propiedades): son los atributos configurables de los componentes visuales (como color, texto, tamaño).

- ***Events*** (eventos): son las acciones que esos componentes pueden generar (clics, movimientos del ratón) que tú manejas con el código.

- ***Code*** (código): es el lenguaje Java que escribes para dar funcionalidad a esos eventos y definir la lógica de tu aplicación, conectando todo lo visual (propiedades) con las acciones (eventos).

Properties (propiedades)

- **¿Qué son?**: son características o atributos de los objetos (componentes) que puedes ver y modificar en el diseñador gráfico de NetBeans.

- **Ejemplos**: el texto de un botón (JButton), el color de fondo de un panel, el tamaño de una ventana, si un campo de texto es editable o no.

- **¿Dónde se configuran?**: en la ventana «propiedades» (*properties window*) de NetBeans, a menudo con métodos *set* y *get* en el código.

Events (eventos)

- **¿Qué son?:** son sucesos o acciones que ocurren en la interfaz gráfica, generalmente, provocados por el usuario (clics, tecleo, mover el ratón).

- **Ejemplos**: ActionListener (cuando se hace clic en un botón), KeyListener (cuando se presiona una tecla), MouseEvent (acciones del ratón).

- **¿Cómo funcionan?**: NetBeans te permite «escuchar» estos eventos y asociarles un método (una parte de tu código) que se ejecutará cuando ocurran.

Code (código)

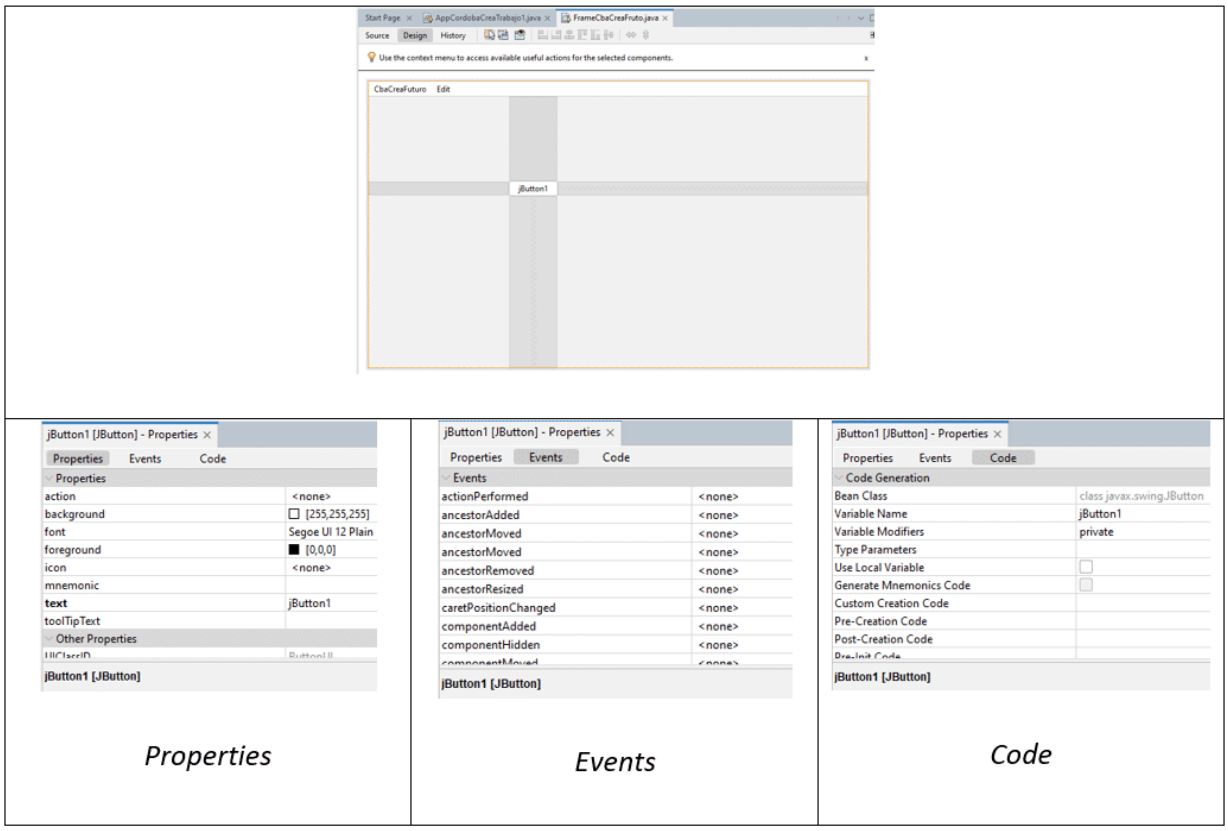
- **¿Qué es?**: es tu programa escrito en Java que define la lógica y el comportamiento de la aplicación.

- **Función**: aquí implementas los métodos que responden a los eventos, manipulas las propiedades de los objetos y creas la funcionalidad general de tu aplicación.

- **Relación con los otros dos**: el código Java es el «pegamento» que toma una acción (evento) y cambia una característica (propiedad) o realiza una tarea.

En resumen, el diseñador para establecer las *properties* de tus componentes, el sistema de eventos para detectar lo que el usuario hace y el *code* (Java) para programar la respuesta a esos eventos, modificando propiedades o ejecutando acciones.

Figura 9: Se ha agregado un botón y se muestran las características del *Palette* del botón



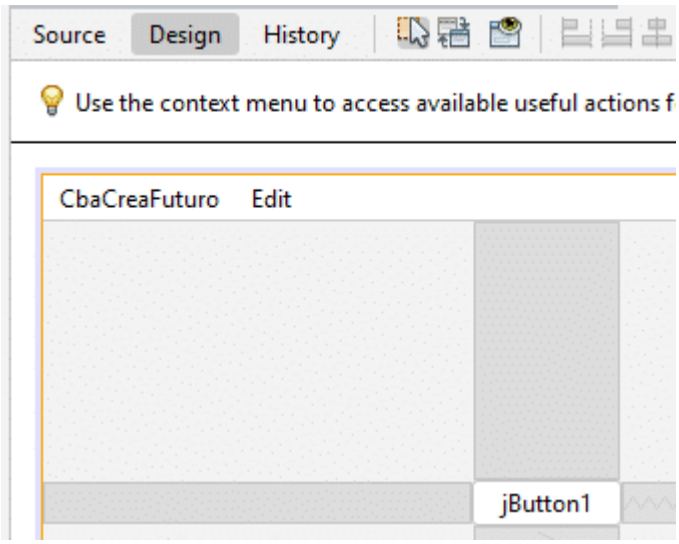
Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Si se observa en la parte de arriba de la figura 9, se podrá ver que tiene tres solapas, que mencionaremos a continuación.

- ***Source*** te permite ver y editar el código fuente directamente.
- ***Design*** te da una vista gráfica para diseñar interfaces (GUI) arrastrando y soltando componentes.

- ***History*** (o historial) te permite acceder a versiones anteriores o cambios realizados en tus archivos, especialmente útil con sistemas de control de versiones. Estas pestañas cambian la forma en que interactúas con tu proyecto: código puro, diseño visual o gestión temporal de versiones.

Figura 9.1: *Source, Design, History*



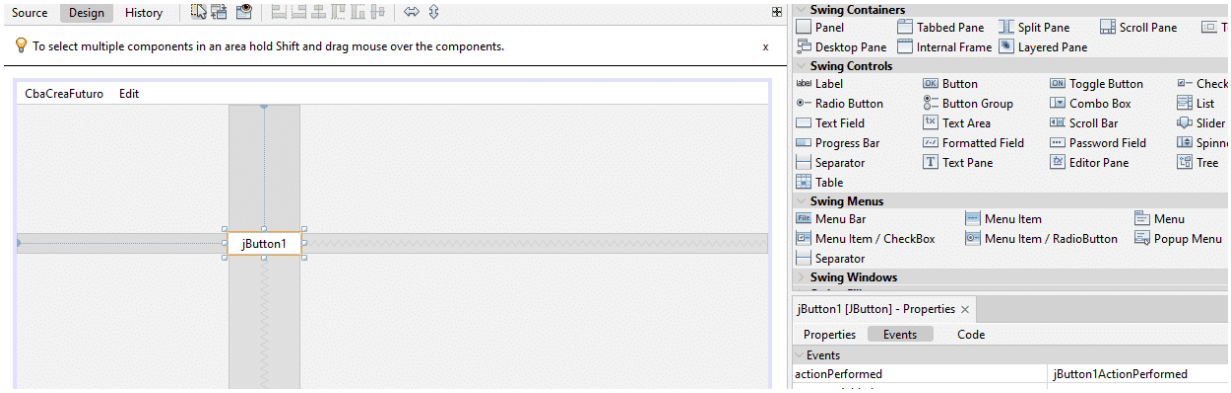
Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Son modos de vista y edición que te permiten trabajar en diferentes niveles: la lógica pura (***Source***), la apariencia visual

(*Design*) y la gestión de cambios a lo largo del tiempo (*History*).

Paso 9: arrastramos un botón, lo seleccionamos y luego seleccionamos **jButton1ActionPerformed** en ***events*** (como observaremos en la figura 10).

Figura 10: Agregamos botón y seleccionamos un evento

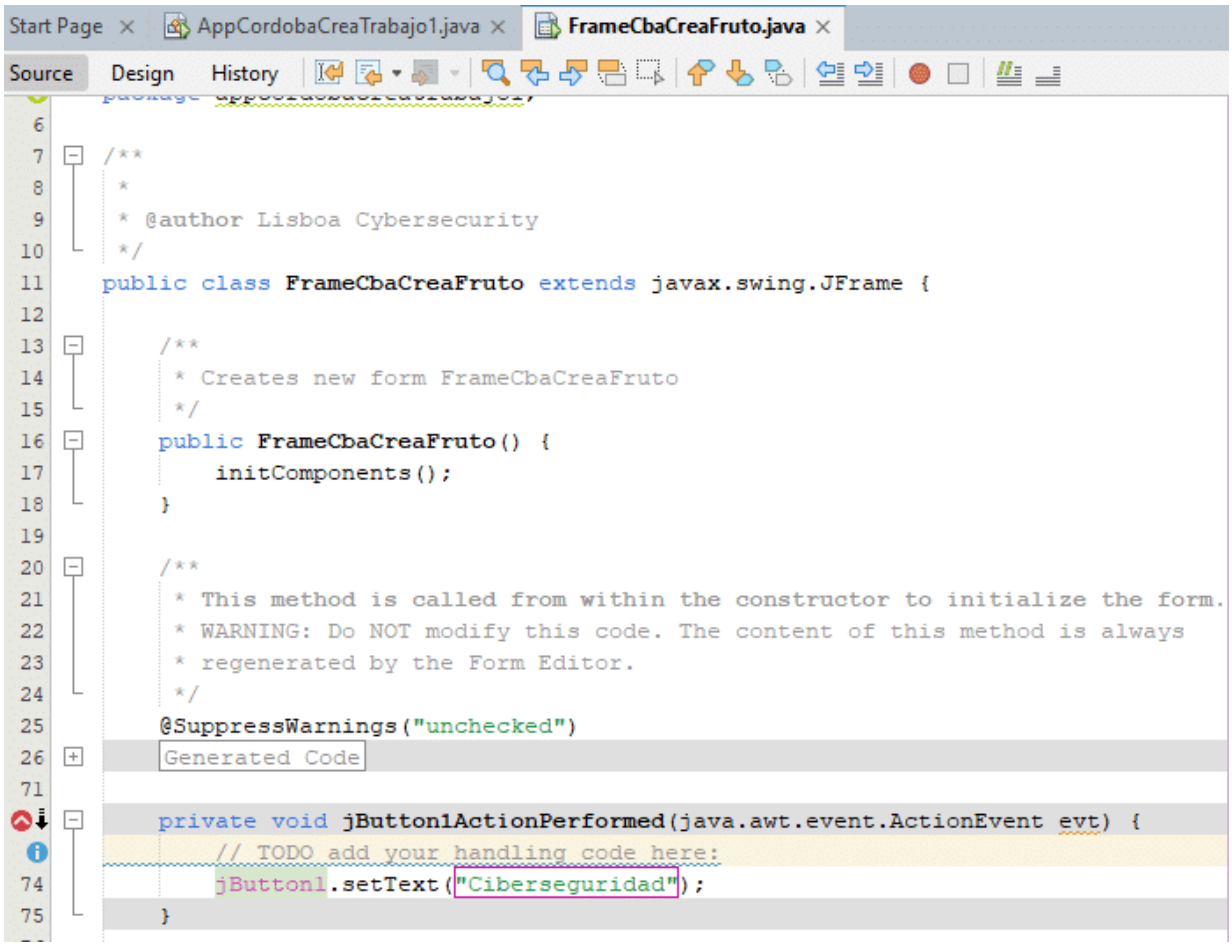


Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Ahora, en tiempo de ejecución, cuando el usuario haga clic en el botón, la leyenda «jButton1» cambiará el texto diciendo la palabra «ciberseguridad». Se destaca que este cambio de leyenda se puede hacer directamente marcando el JButton1 y luego modificando el texto en ***text*** de la solapa ***properties***, pero aquí se hará en tiempo de ejecución agregando el código: **jButton1.setText(«Ciberseguridad»)**.

Para ello, luego de seleccionar **jButton1ActionPerformed** en ***events*** (figura 10), automáticamente se muestra lo que se visualiza en la imagen 10.1 y se agrega el código mencionado en el párrafo anterior.

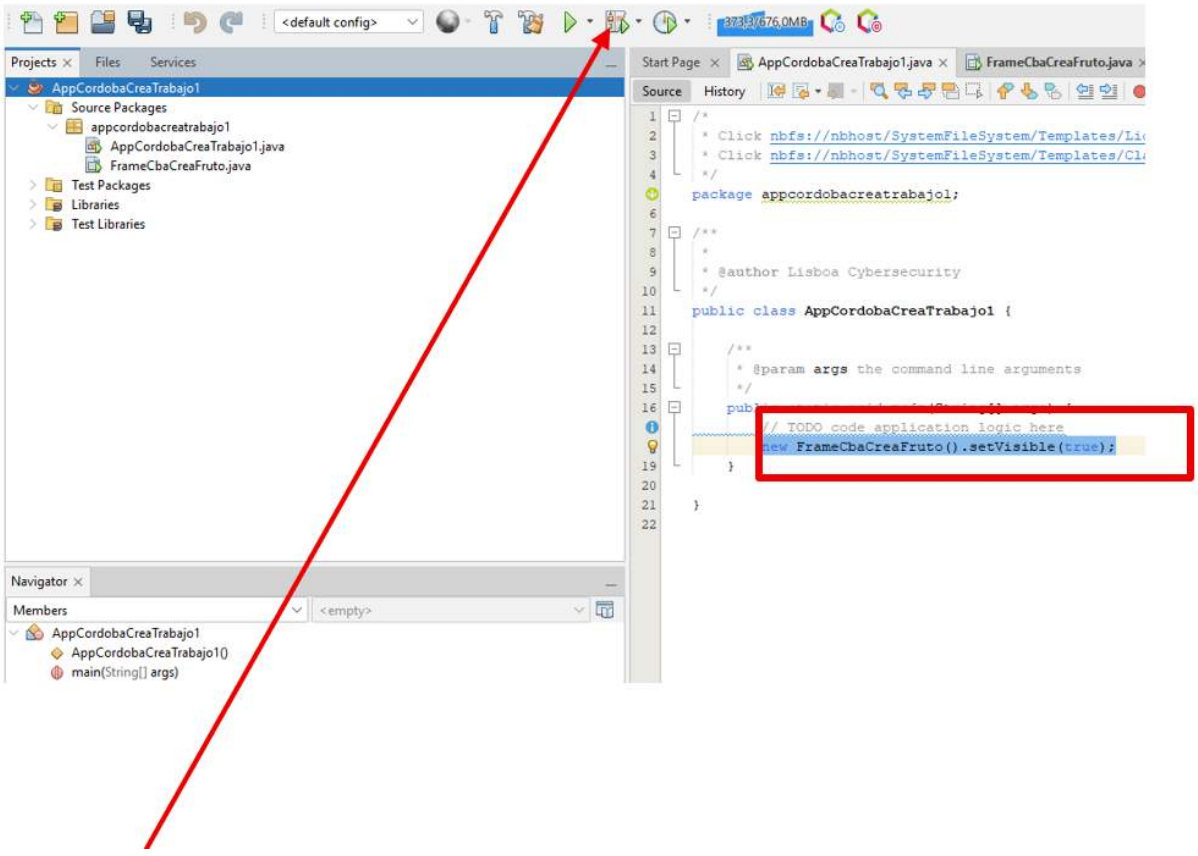
Figura 10.1: Agregando código jButton1.setText("Ciberseguridad")



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Antes de ejecutar el código, se recomienda un último cambio: en el AppCordobaCreaTrabajo1, se agrega el siguiente código: *new* **FrameCbaCreaFruto().setVisible(true);** para que invoque al JFrame que contiene el menú y el botón recién agregado (FrameCbaCreaFruto).

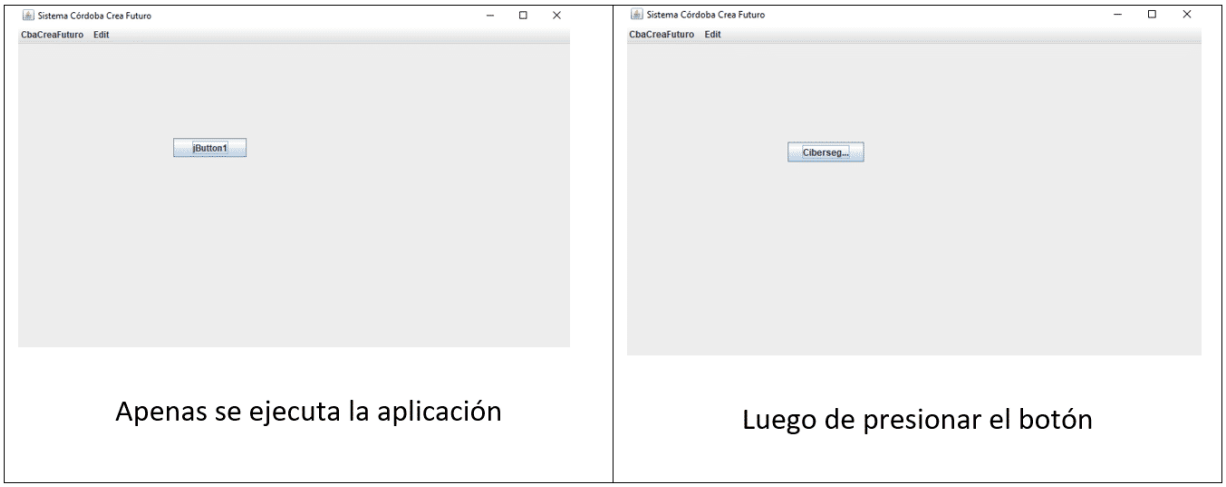
Figura 10.2: Agregando código *new* FrameCbaCreaFruto().setVisible(true)



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Ahora se puede ejecutar el programa y probar presionando el botón para visualizar el cambio mencionado.

Figura 10.3: Ejecutando el programa o aplicación desarrollada



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

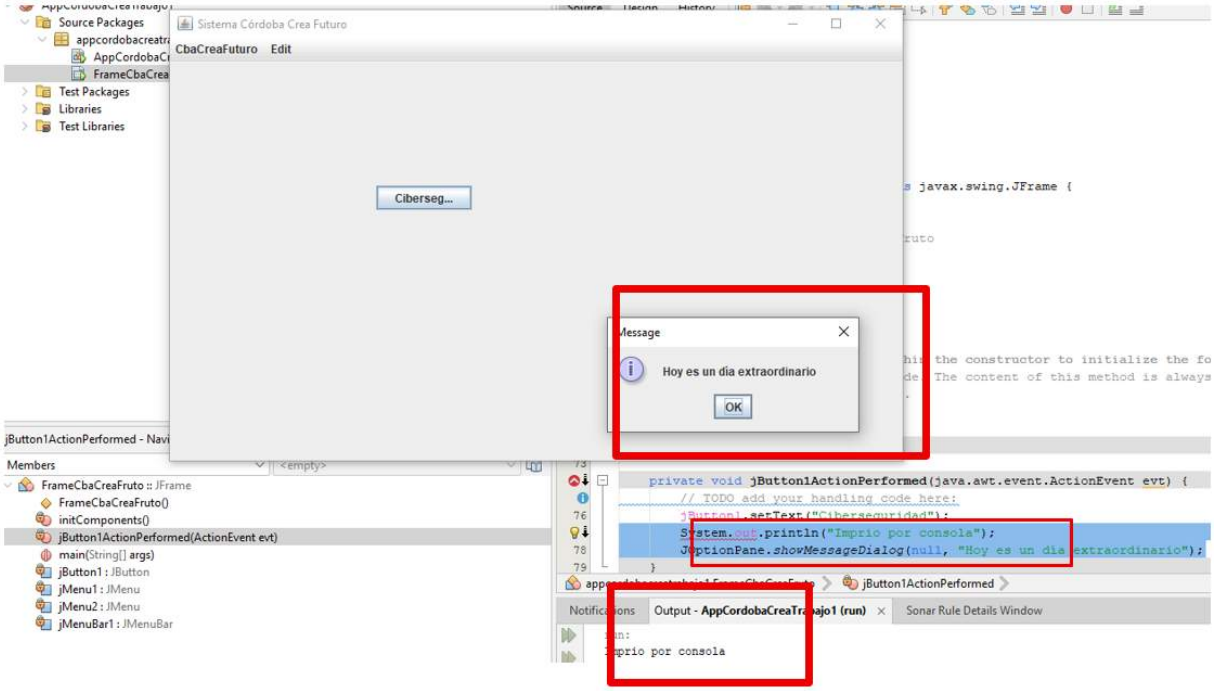
Nota: agregar debajo del código fuente que se muestra en la figura 10.1 las siguientes dos líneas.

1. **System.out.println(«Imprimo por consola»)**.

2. **JOptionPane.showMessageDialog(null, «Hoy es un día extraordinario»).**

Luego ejecutar la aplicación y verificar si, luego de presionar el botón que cambia su texto, también lleva a cabo dos acciones más mostradas en la figura 10.4.

Figura 10.4: Imprimiendo por consola mensaje y otro mediante caja



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

CONTINUAR

3. Agregando elementos

3. Agregando elementos

Como se ha observado, es posible agregar muchos más elementos, los que, según la necesidad del desarrollador, son los que utilizará en su implementación.

Es necesario mencionar que, cualquiera sea la versión de Netbeans o se utilice otra IDE distinta, la *Palette* tiene una filosofía de trabajo similar, es decir, arrastrar y configurar los elementos mediante ***Properties*** y ***Events***.

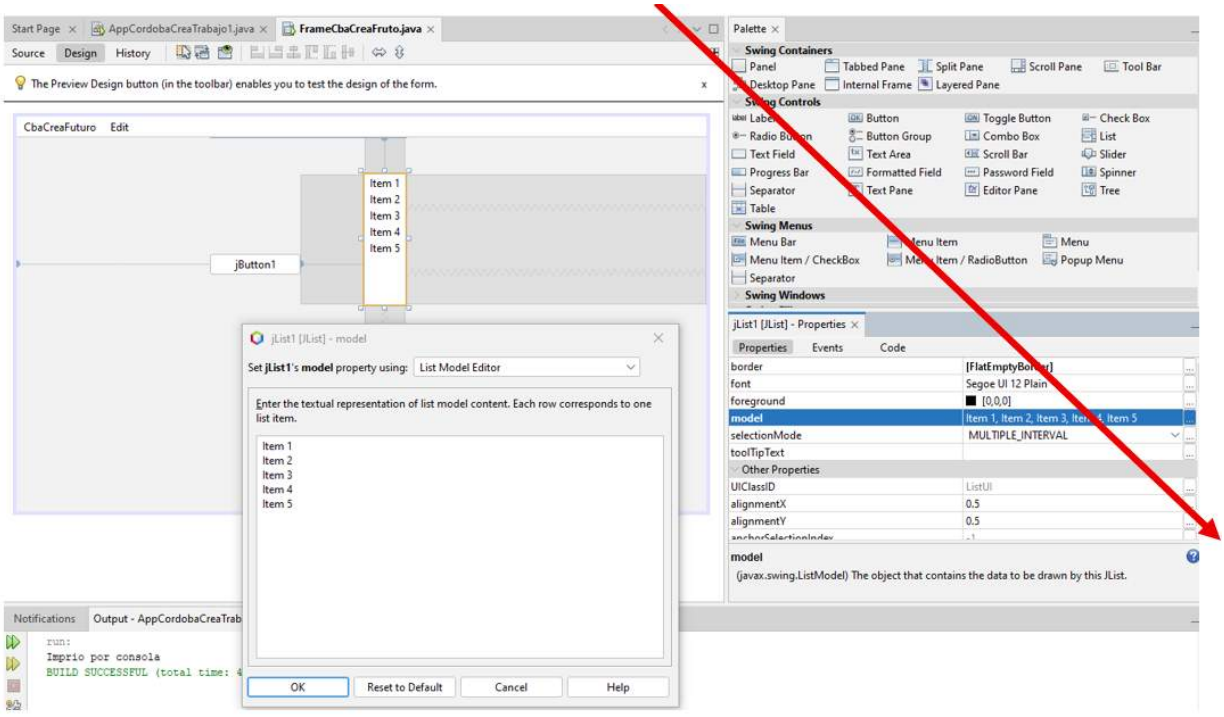
A continuación, se mostrarán algunos ejemplos más y se le sugiere al alumno practicar y usar ***vide coding*** para más ejemplos.

Lista

Arrastramos del ***Palette*** una ***List*** cuyos textos, por defecto, van en forma escalonada: ítem 1, ítem 2, etc. Luego, en ***model***

presionamos los tres puntitos para cambiar esos textos que se visualizan por Córdoba, Río de Janeiro, Seúl, Madrid, Roma.

Figura 11: Agregando una *list*



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

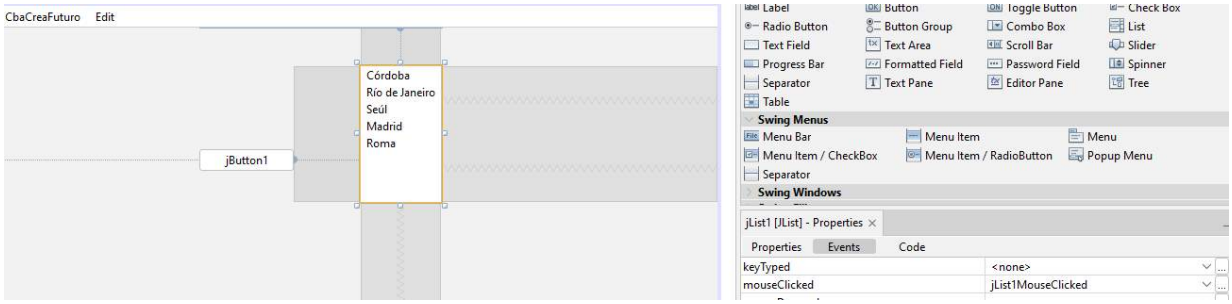
Una vez cambiados a los nombres de ciudades, hacer clic en **mouseClicked** de ***event*** para generar el evento que cada vez que se seleccione un elemento de la ***list***, se ejecute este evento. La idea será que imprima mediante una caja el texto seleccionado, en nuestro caso, la ciudad elegida.

Una vez que se haga clic en el evento mencionado, agregar la siguiente línea de código:

JOptionPane.showMessageDialog(null,"Elegí: "+jList1.getSelectedValue());

donde la opción elegida es entrega el método **jList1.getSelectedValue()).**

Figura 12: Textos de la caja cambiados y generado el evento en mouseClicked



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

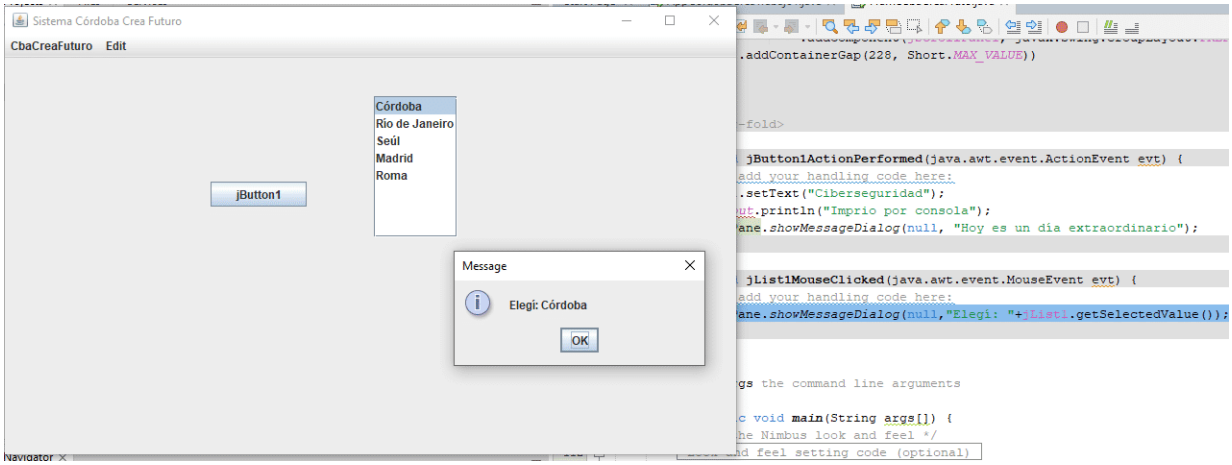
En el apartado ***Source***, se puede ver la línea de código agregada.

Figura 13: Línea de código agregada (ver la resaltada en azul)

Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Si se ejecuta la aplicación y se selecciona un elemento de la lista, el texto de esa elección se imprime en una caja, como se puede ver en la figura 14.

Figura 14: Imprime el texto seleccionado

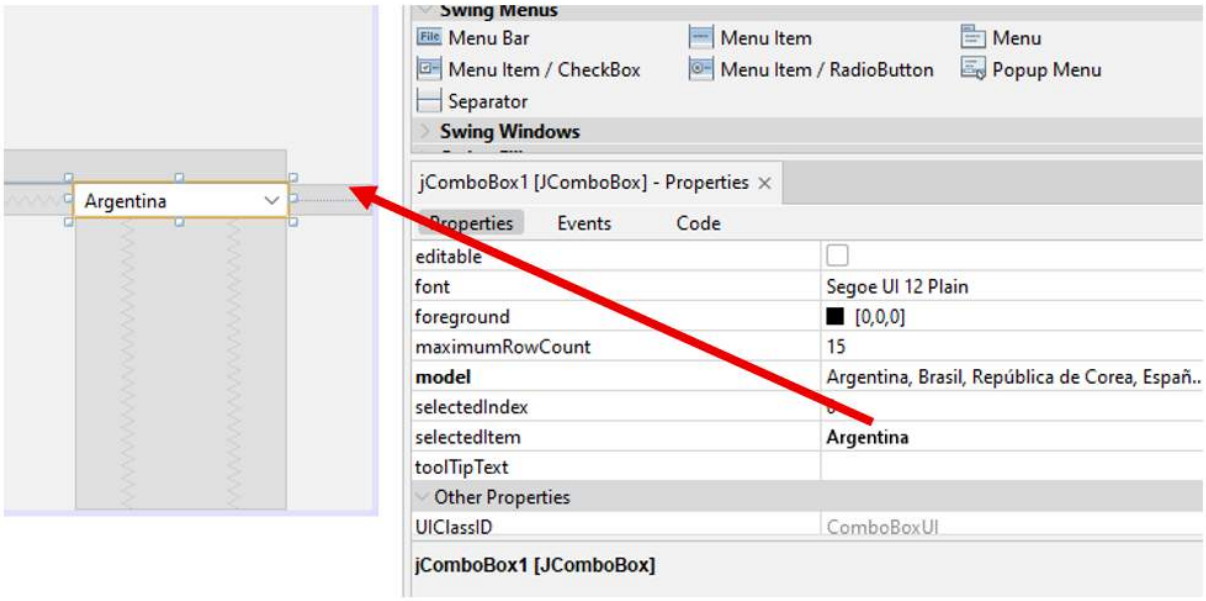


Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Combo Box

Una situación similar a ***list*** es **combo *box*,** el cual arrastraremos desde el ***Palette*** y luego cambiaremos los ítems por defecto por los siguientes países: Argentina, Brasil, República de Corea, España, Italia. El texto que inicialmente se muestra en la caja está dado por el selectedIndex que inicia en cero (0) para indicar que es el primer elemento.

Figura 15: Selección de elemento a mostrar en el Combo *Box*



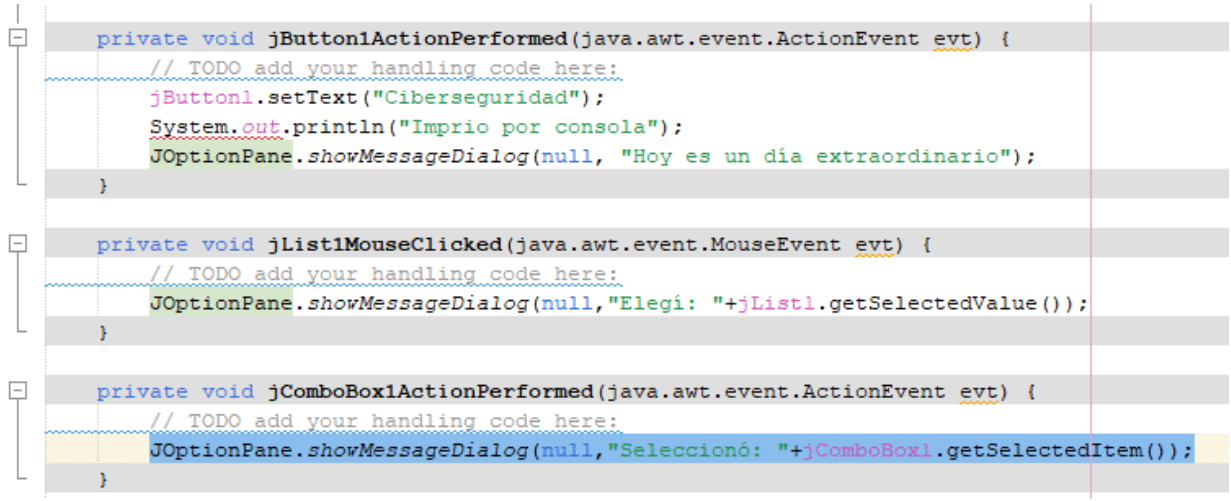
Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Lo que se hará seguidamente es que, una vez seleccionado un elemento (país), el selecedIndex cambia y, usando el selectedItem que inicialmente marca el país Argentina, se podrá obtener el país elegido.

Para lograrlo, hacer clic en **actionPerformed** de ***events***, el que genera el método **jComboBox1ActionPerformed** que permitirá imprimir la opción elegida mediante el agregado de la siguiente línea de código:

JOptionPane.showMessageDialog(null,«Seleccionó: »+jComboBox1.getSelectedItem()).

Figura 16: Se agrega la sentencia resaltada con azul

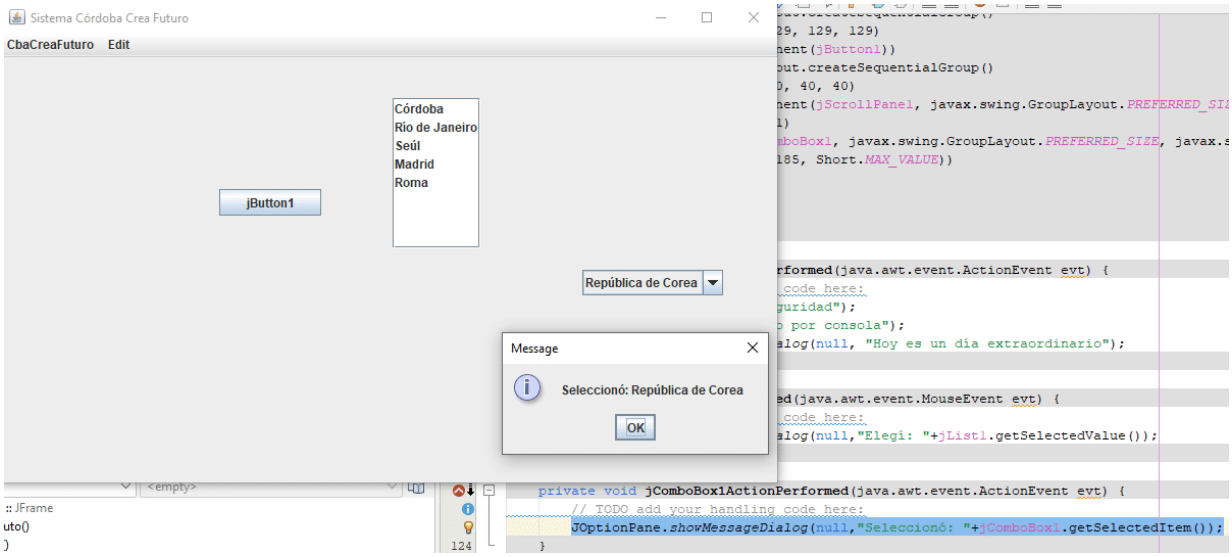


```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jButton1.setText("Ciberseguridad");
        System.out.println("Imprio por consola");
        JOptionPane.showMessageDialog(null, "Hoy es un día extraordinario");
    }

    private void jList1MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        JOptionPane.showMessageDialog(null,"Elegí: "+jList1.getSelectedValue());
    }

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        JOptionPane.showMessageDialog(null,"Seleccionó: "+jComboBox1.getSelectedItem());
    }
```

Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Ejecutando la aplicación, se muestra en un mensaje el país seleccionado, como lo muestra la figura 17 a continuación.

Figura 17: Muestra en un mensaje el país seleccionado



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

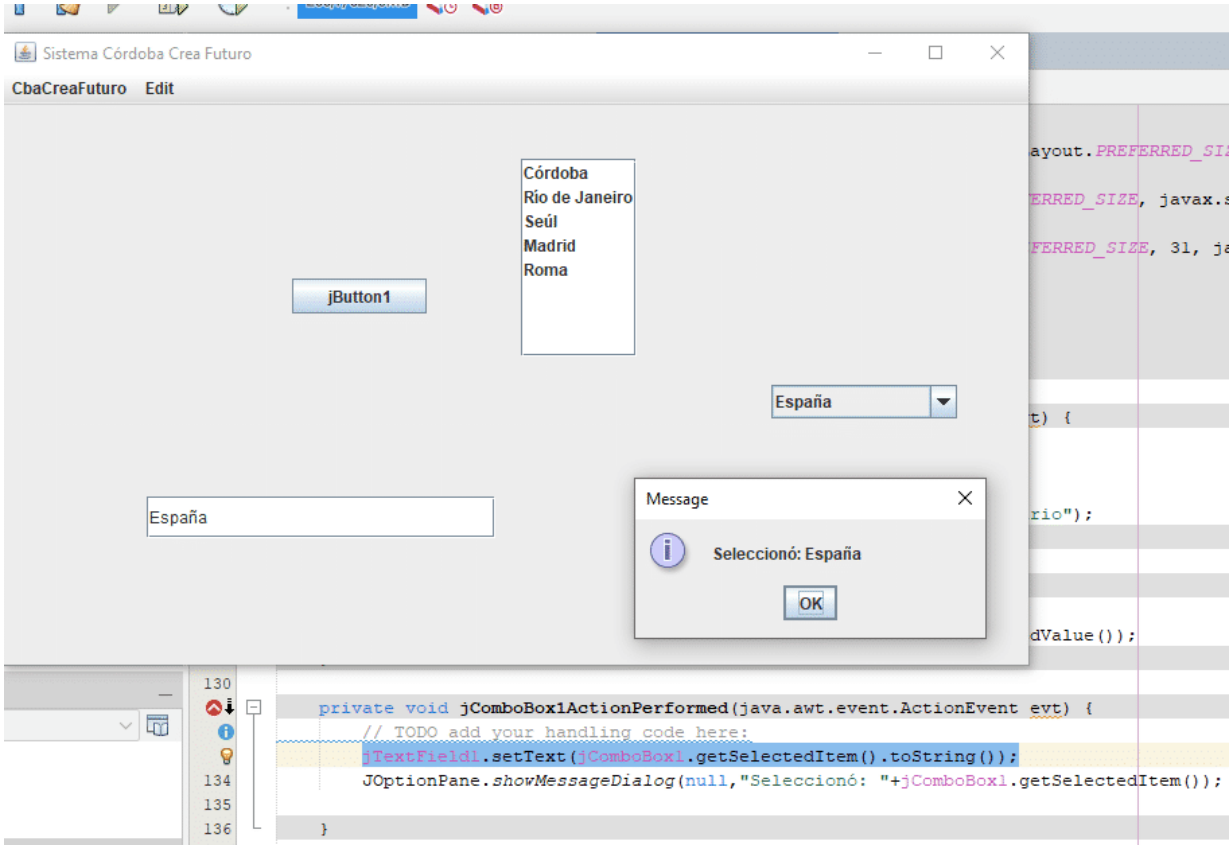
Text field

Para explicar el uso de esta caja que permite ingresar un texto, dato, etc., se escribirá automáticamente el país que se haya seleccionado. Para ello, se debe arrastrar un *text field* al marco de trabajo que se viene desarrollando, y luego en **jComboBox1ActionPerformanced** del ***events*** de **combo *box*** se debe agregar la siguiente línea de código:

```
jTextField1.setText(jComboBox1.getSelectedItem().toString());
```

Ejecutando la aplicación, se verá del mismo modo en que se observa en la siguiente figura.

Figura 18: Se agregó una línea de código y se ejecuta



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

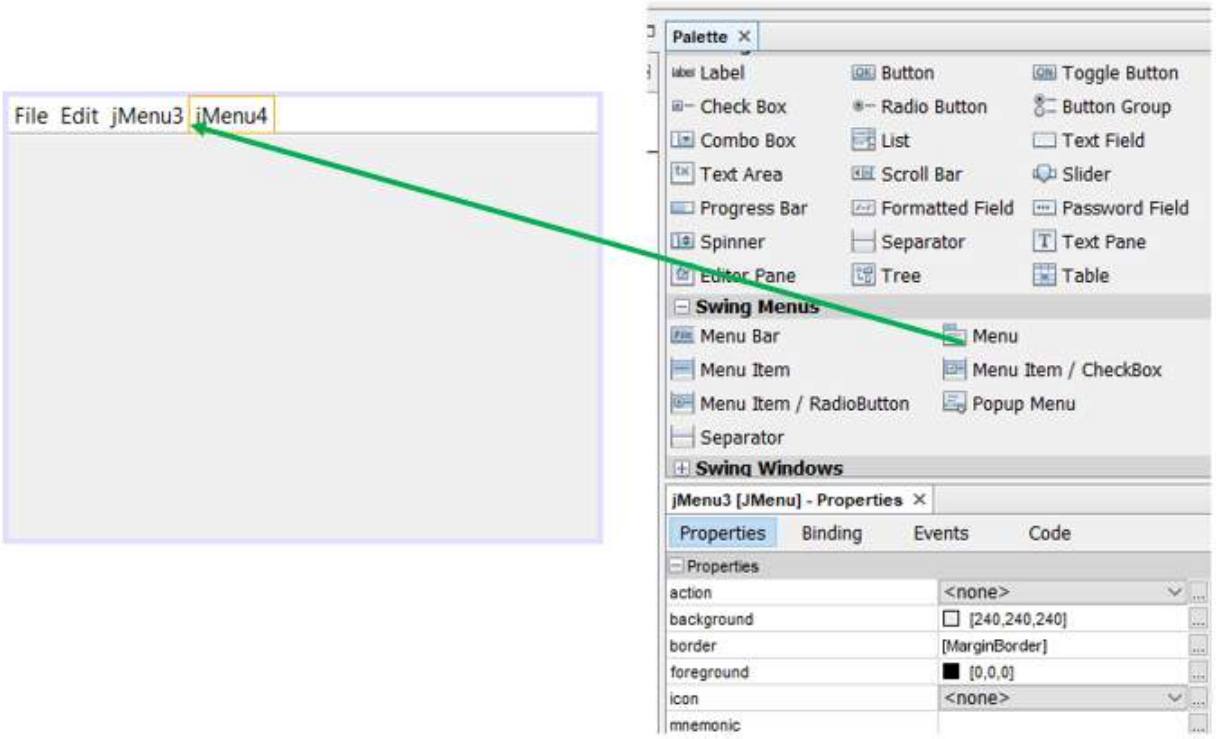
Más de menú

Las siguientes imágenes son significativas y permiten visualmente entender la facilidad de armar los menús de opciones.

Seguidamente, les comparto imágenes que el autor de este módulo desarrolló para los alumnos de una cátedra en universidad.

Arrastramos dos veces seguidas **menú** y tendremos jMenu3 y JMenu4.

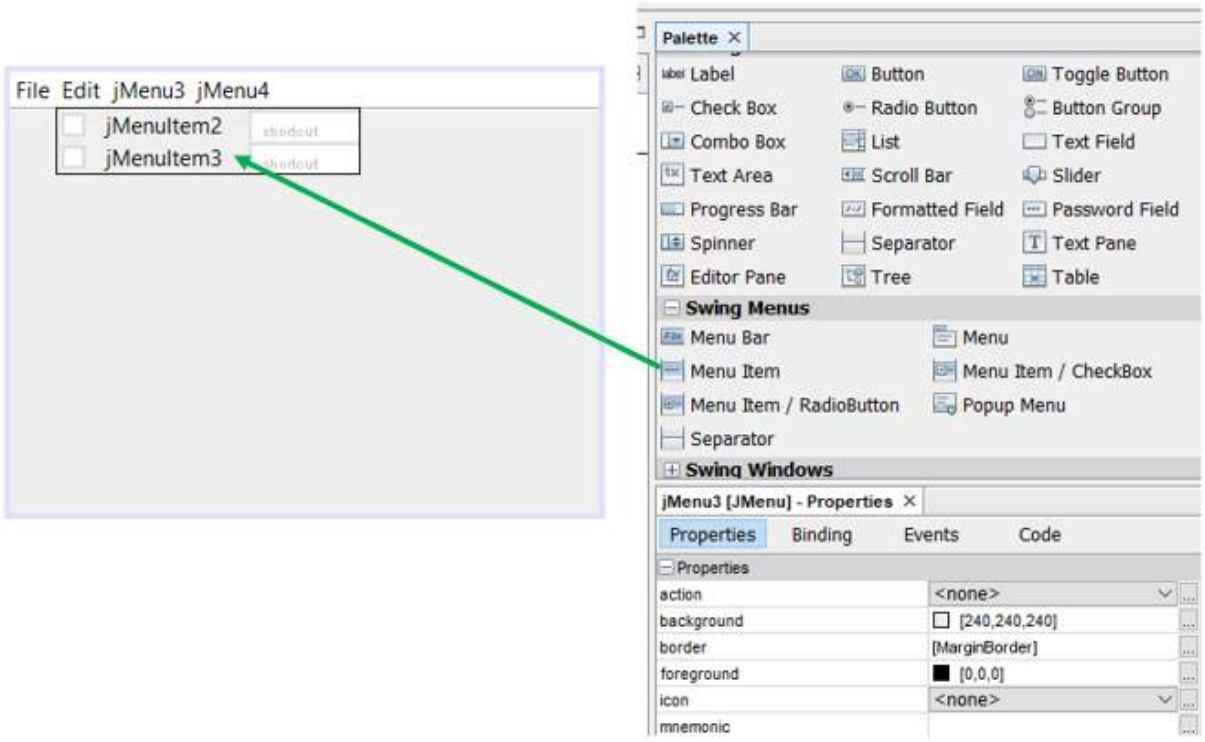
Figura 19: Agregando menús



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Arrastramos dos MENU ITEM al primer componente del menú; en este ejemplo sería a *FILE*.

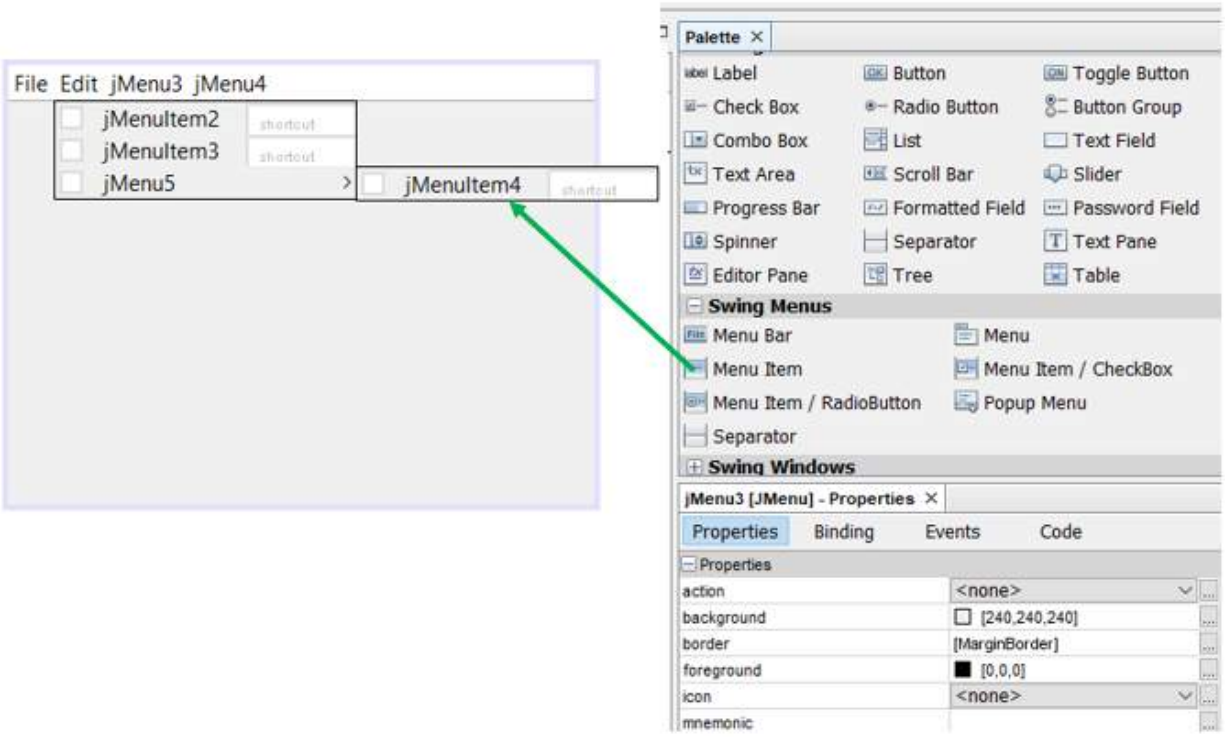
Figura 20: Agregando menú ítem



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Arrastramos un **menú** más y a este le arrastramos un **menú ítem**.

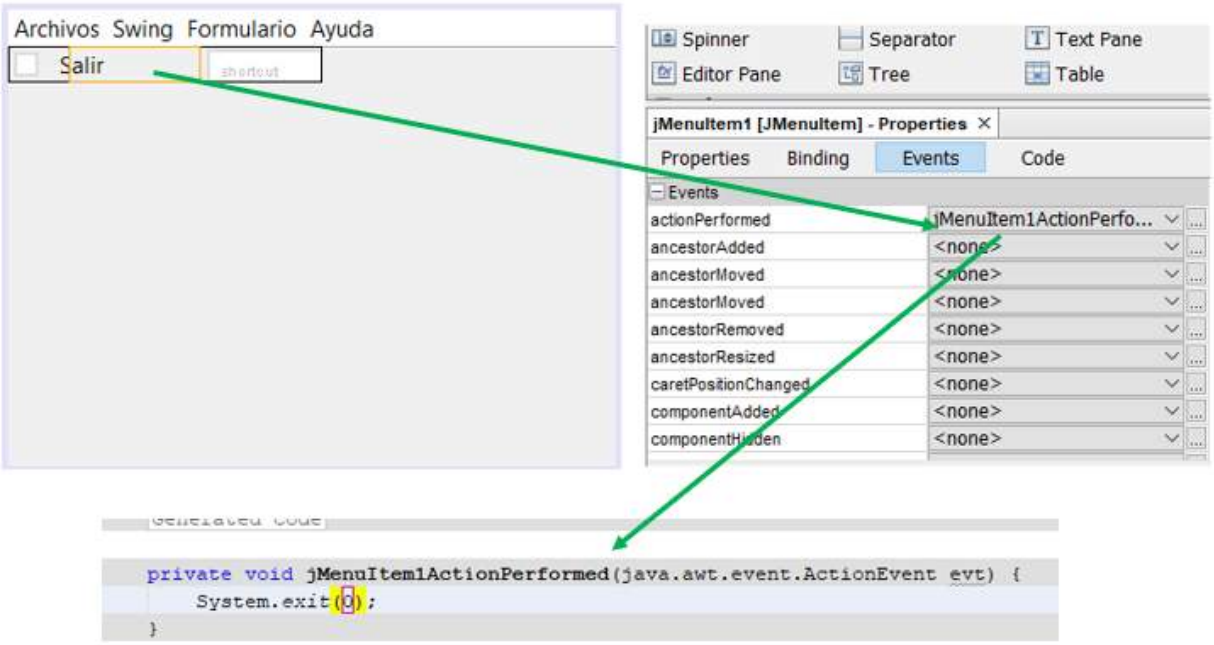
Figura 21: Agregando complejidad



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

Si una opción tuviese la leyenda «salir», para indicar que efectivamente termina la ejecución del programa que estamos ejecutando, en el evento **actionPerformanced** de ***events*** debe agregarse el código **System.exit(0).**

Figura 22: Agregando salida al programa como una opción más



Fuente: elaboración propia con base en captura de pantalla de Apache NetBeans IDE 21.

CONTINUAR

Descarga en PDF