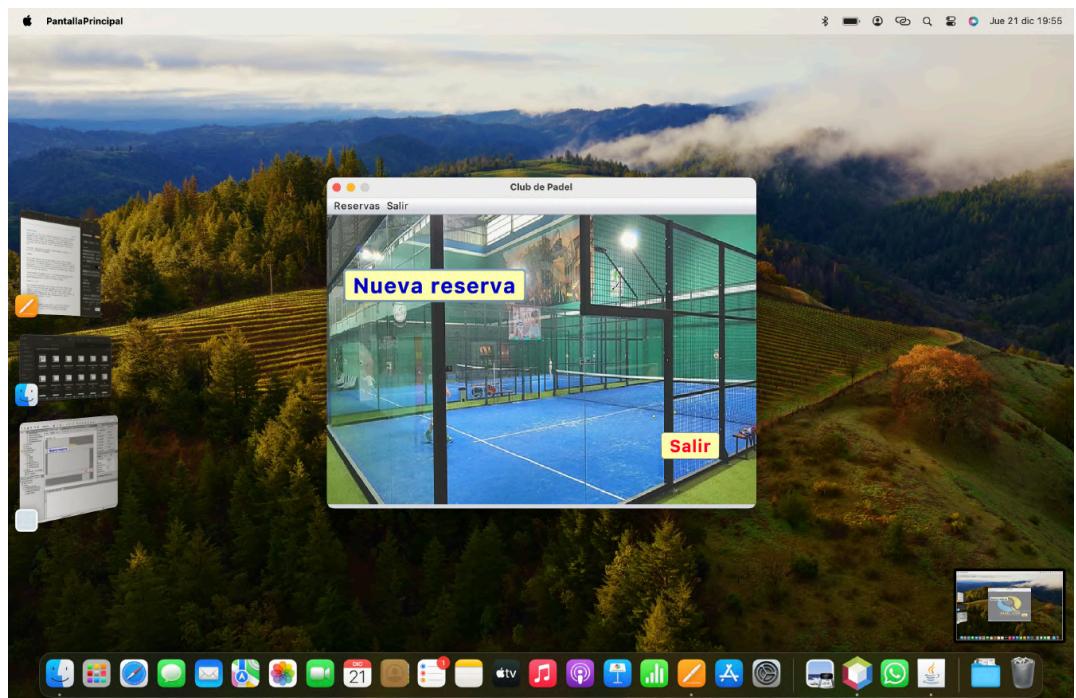
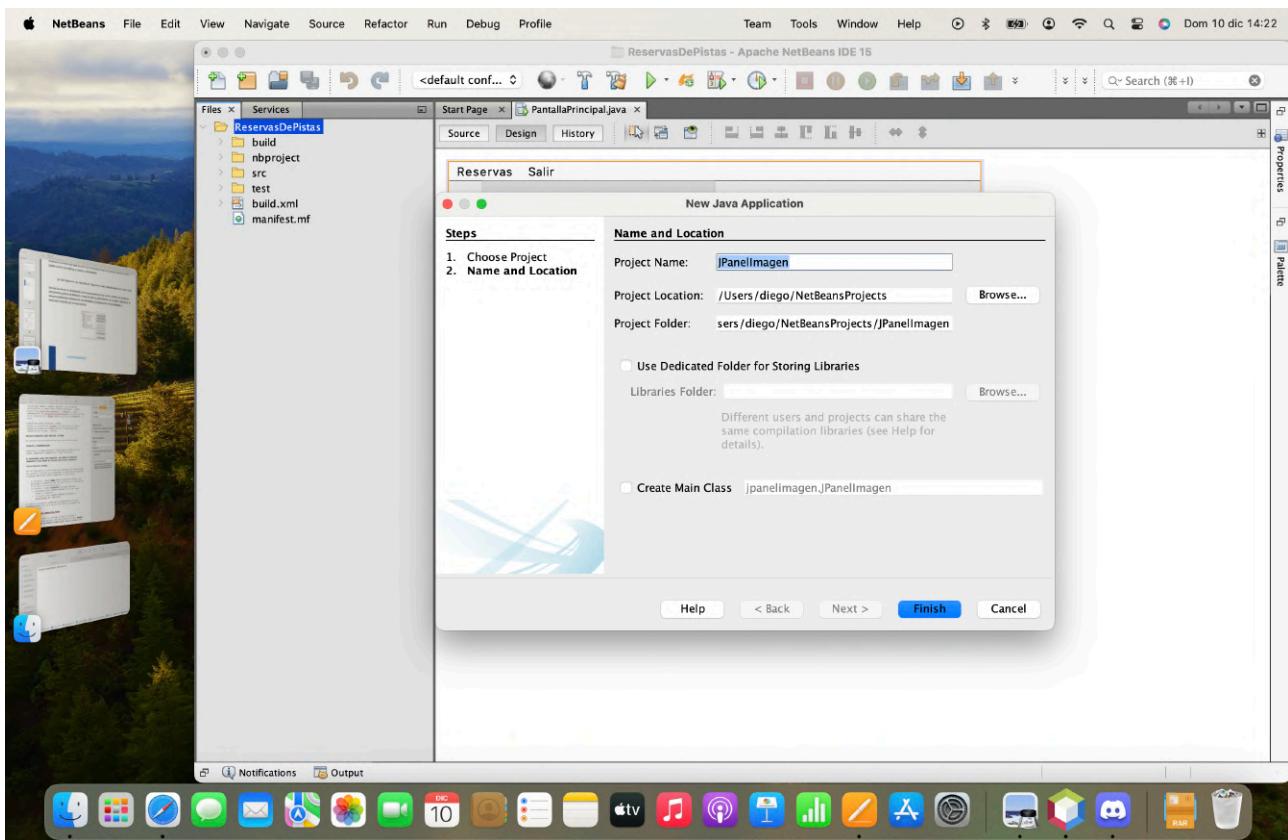


Tarea 3 para Desarrollo de Interfaces

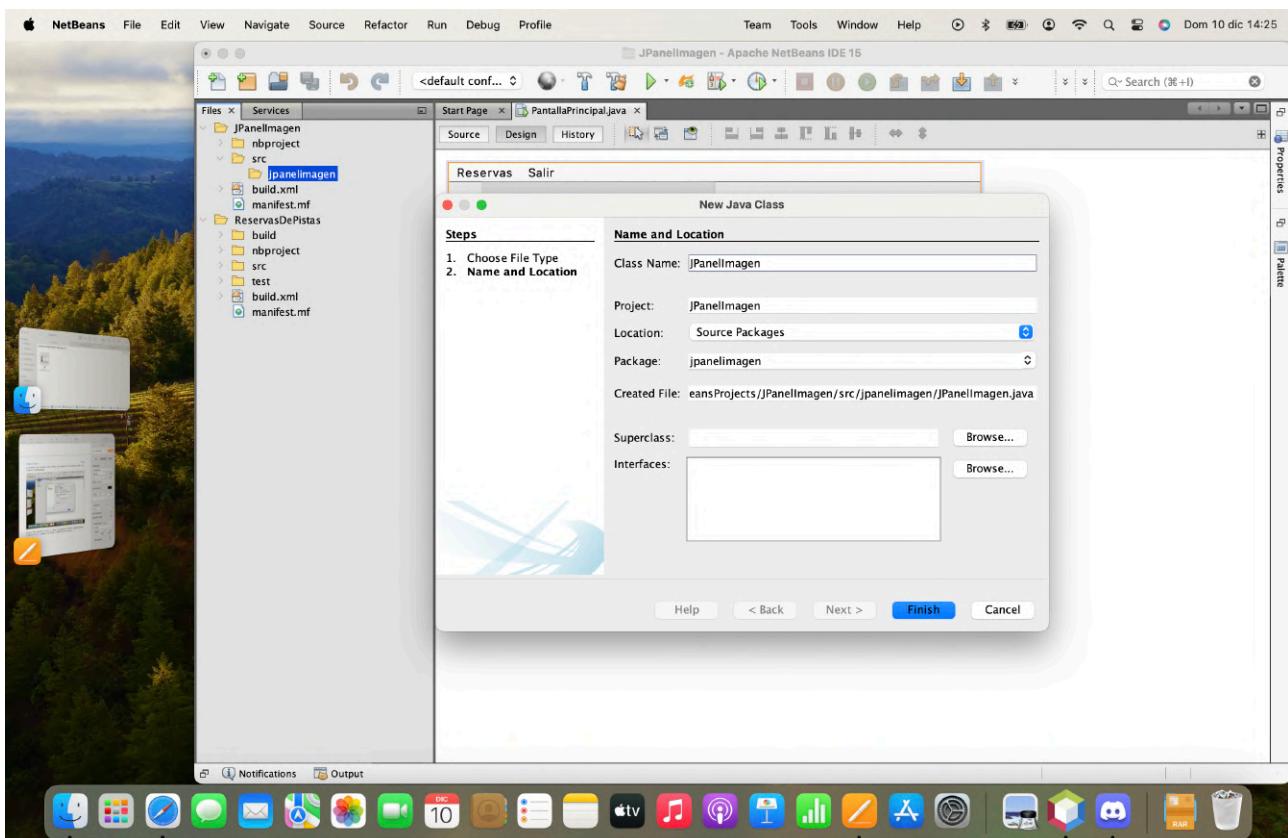


Diego Manuel Carrasco Castañares

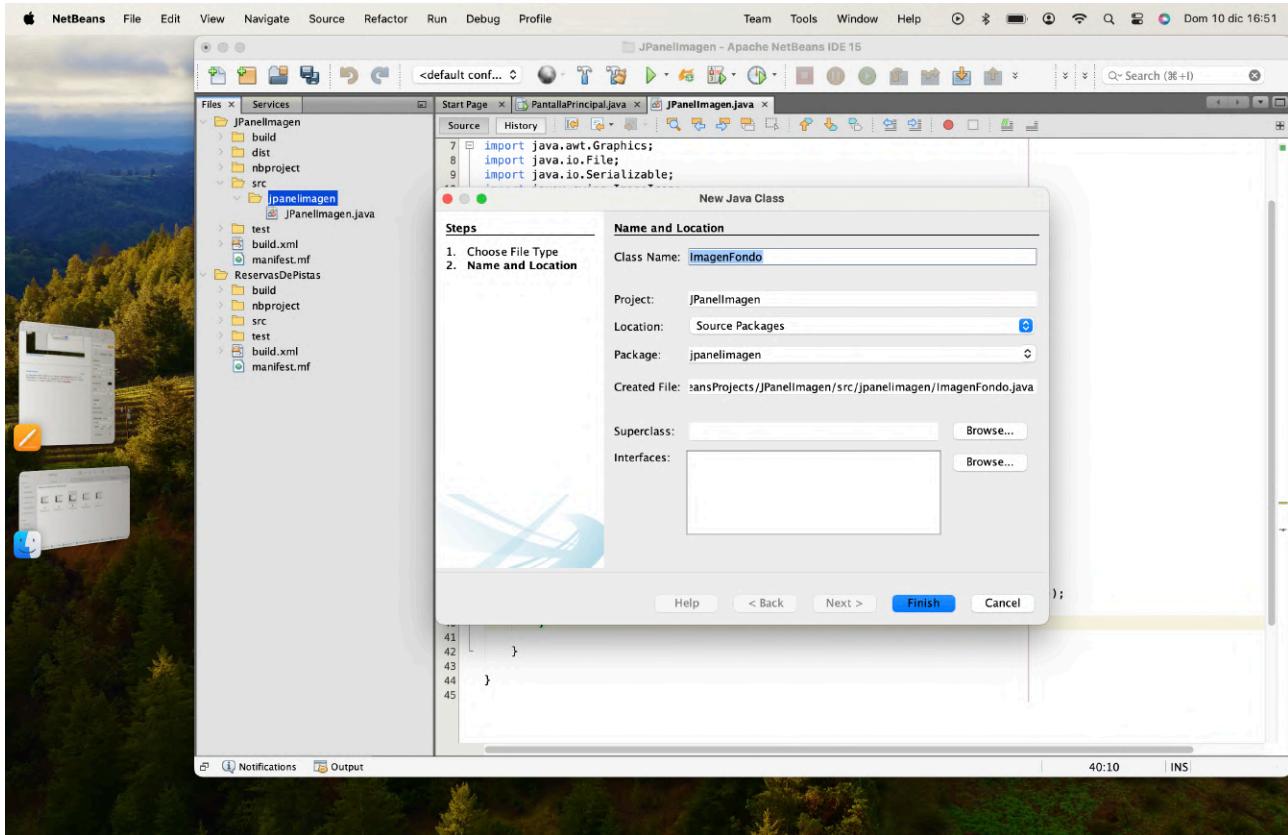
Lo primero que haremos será crear el proyecto. En nuestro caso se llamará JPanelImagen.



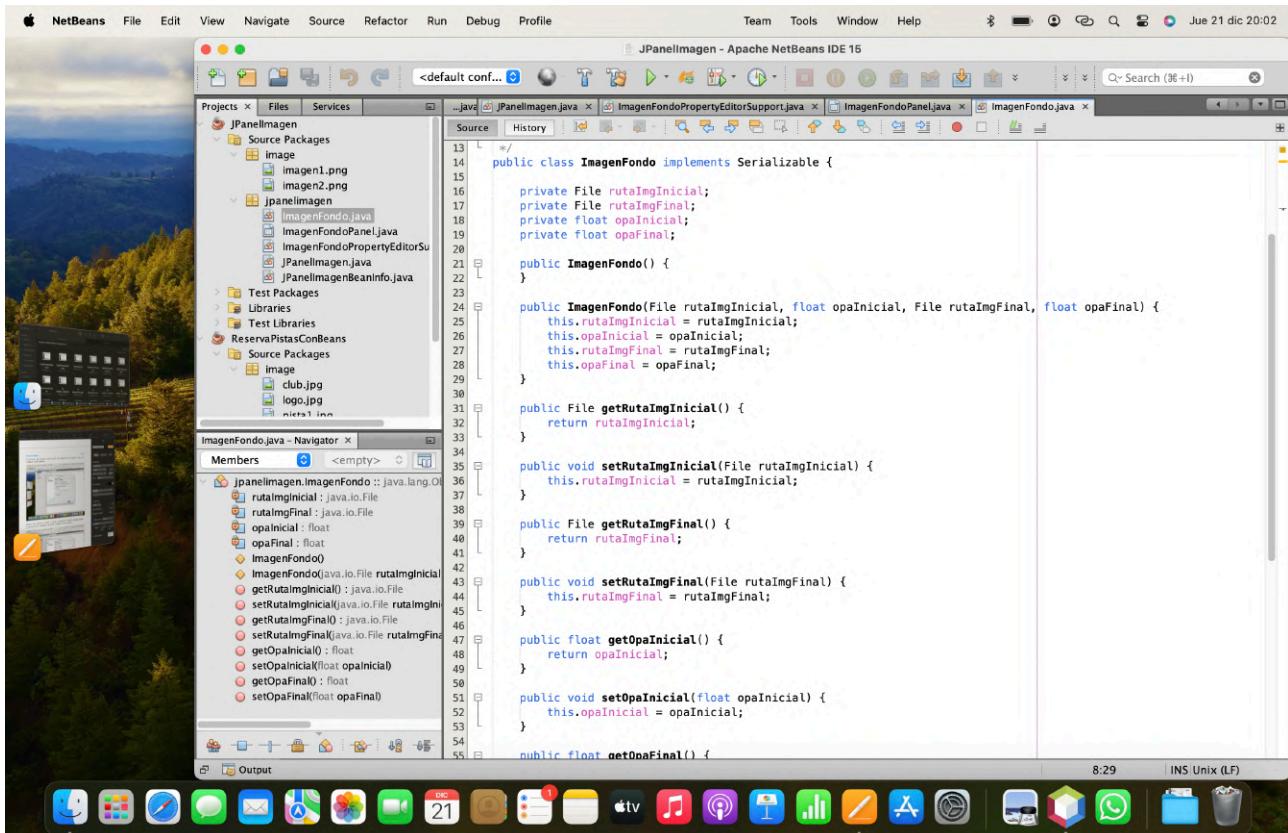
Dentro del proyecto vamos a crear un paquete llamado `jpanelImagen` y dentro de él creamos la clase llamada `JPanelImagen`.



Lo siguiente que haremos será crear una clase llamada ImagenFondo.



Esta clase contendrá cuatro atributos privados, dos de tipo File llamado rutaImgInicial y rutaImgFinal y otros dos de tipo float llamado opaInicial y opaFinal, el constructor por defecto, un constructor con sus cuatro parámetros y los getter and setter.



Volviendo a la clase JPanelImagen, dentro de ella, le diremos que extiende de JPanel y que es serializable.

Este clase tendrá cuatro atributos privados, uno de tipo ImagenFondo, dos de tipo boolean llamados ratonPresionado y contador iniciados a false y uno de tipo Point llamado puntoPresion.

Crearemos un constructor por defecto y dentro de el crearemos un método addMouseListener. Dentro de dicho método sobreescribiremos el método mouseRelease. Dentro de mouseRelease crearemos un if, el cual si se ha presionado el ratón y arrastrado en mas de 50 pixeles con el presionado (que comprobaremos mediante la clase Matt.abs), cambiara el contador de true a false o de false a true, segun se encuentre, y llamara al método repaint. Posteriormente ratonPresionado se pasara a false.

También sobreescrbiremos el método mousePressed para cambiar la variable ratonPresionado a true y guarde el punto donde ha sido presionado dentro de la variable puntoPresion

También crearemos el constructor por defecto, los getter and setter, un atributo privado de tipo ImagenFondo (clase creada anteriormente) llamado imagenFondo, otro de tipo boolean que inicializaremos a false llamado ratonPresionado, otro de tipo boolean llamado contador que también inicializaremos a false y redefiniremos el método paintComponent.

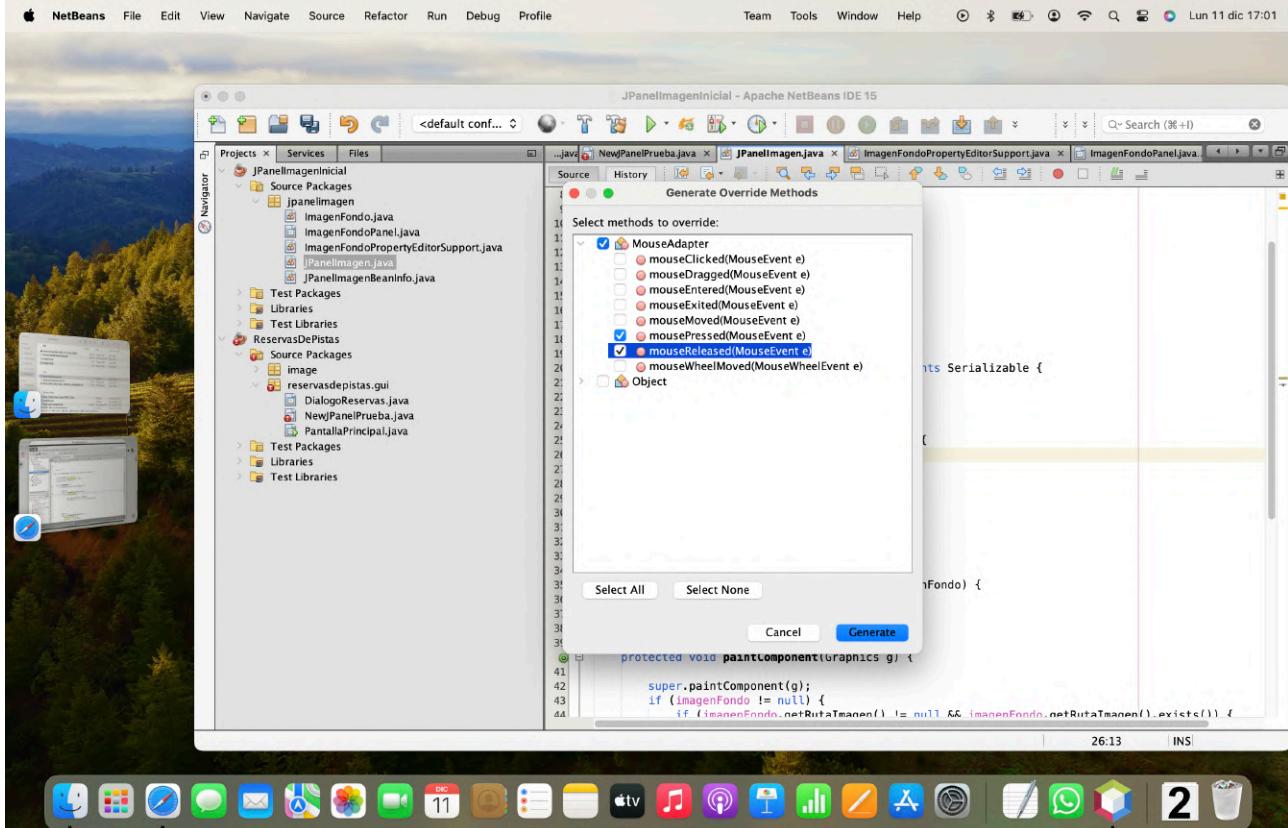
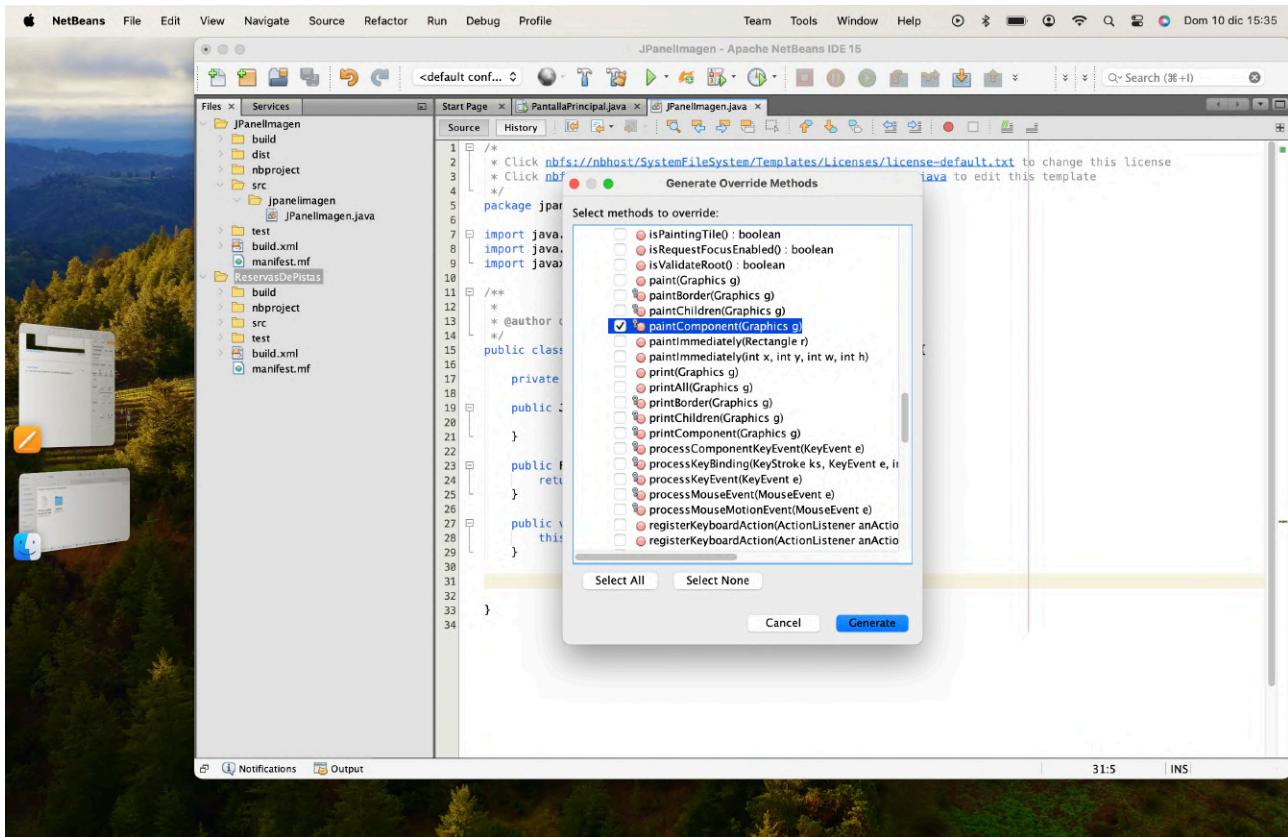
En dicho método llamaremos al padre, comprobaremos si imagenFondo es distinto de null mediante un if, dentro de dicho if declararemos dos variables, una de tipo file llamada ruta mostrada y otra de tipo float llamada opaMostrada y por ultimo declararemos un segundo if.

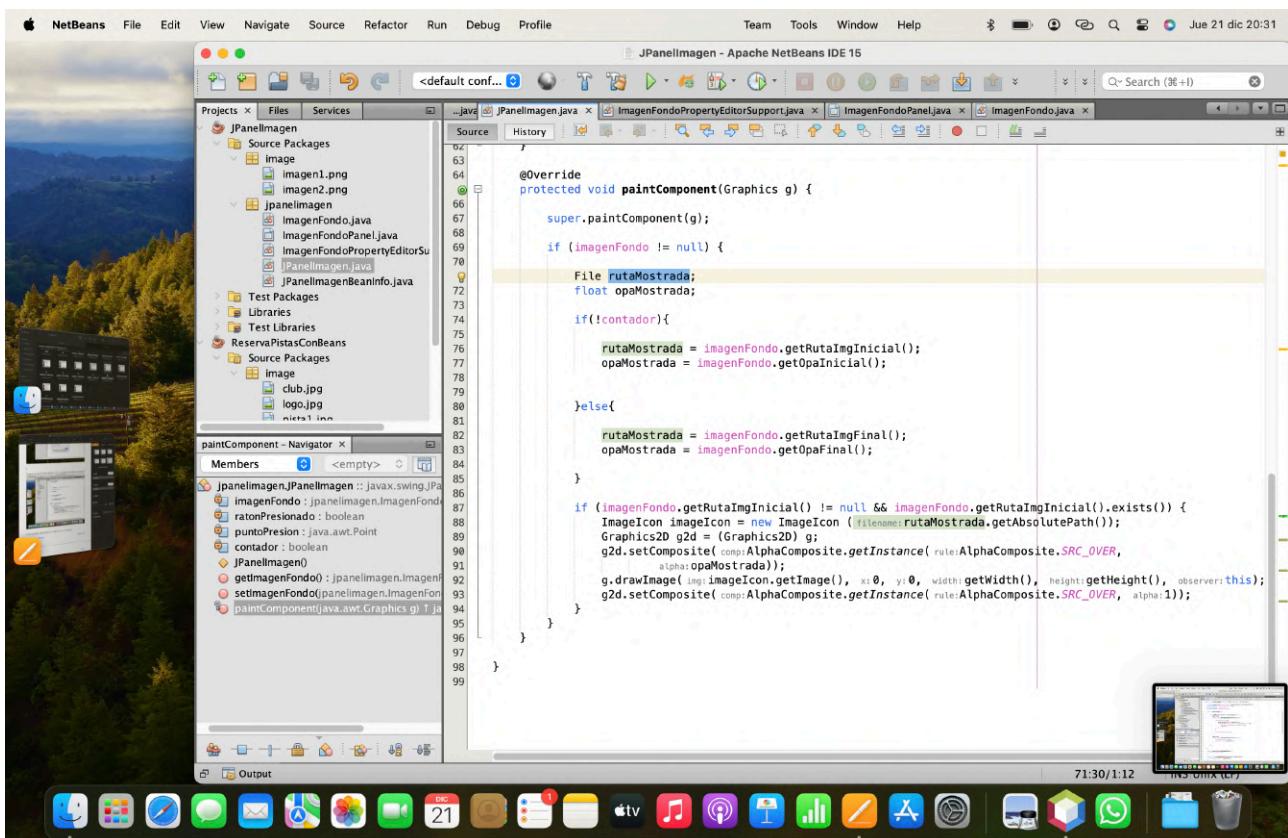
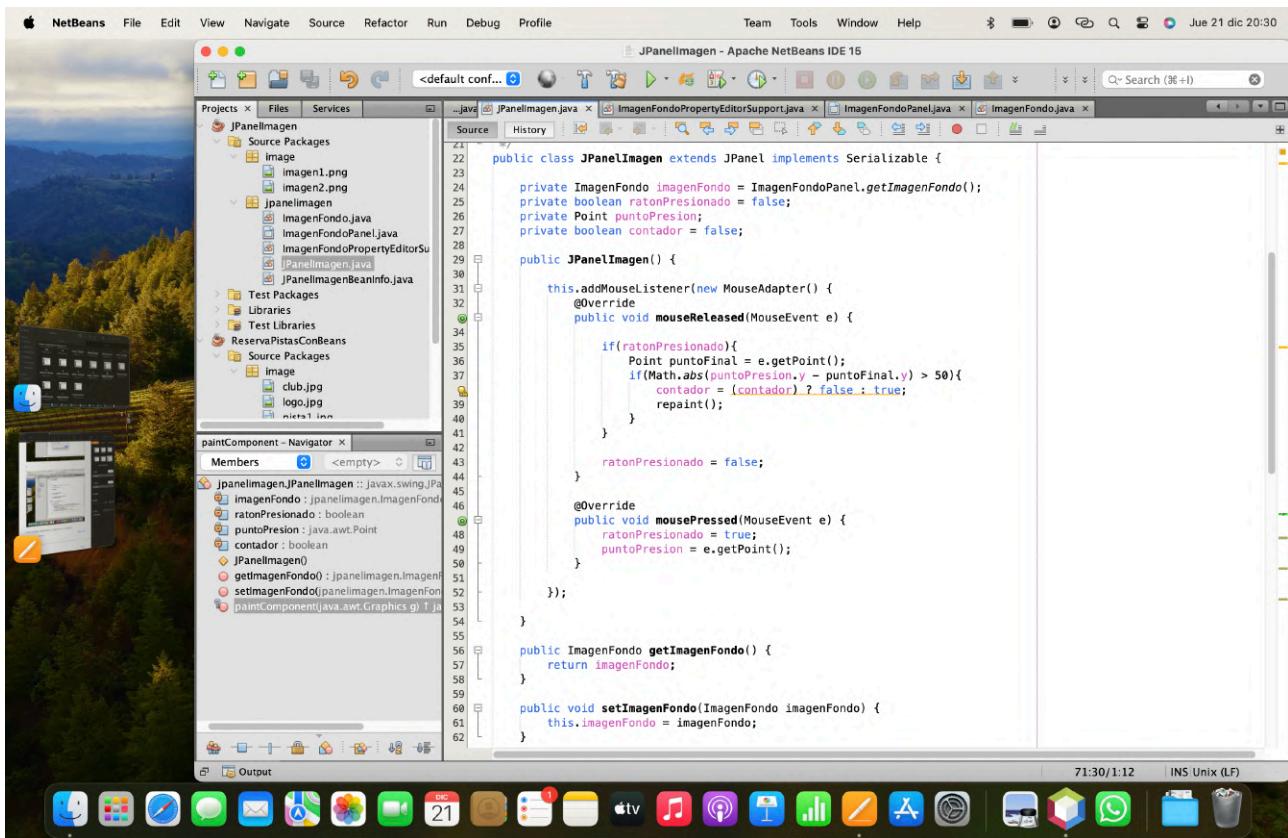
Este segundo if comprobara si la variable contador es false, y si es así instancia en rutaMostrada la ruta imagen inicial y en opaMostrada la opacidad de la imagen inicial. Por el contrario, si la variable contador es true, instancia en las variables anteriormente mencionadas la ruta y la opacidad de la imagen final.

Posteriormente crearemos un tercer if dentro del primero para que nos dibuje la imagen mediante la clase Graphics2D.

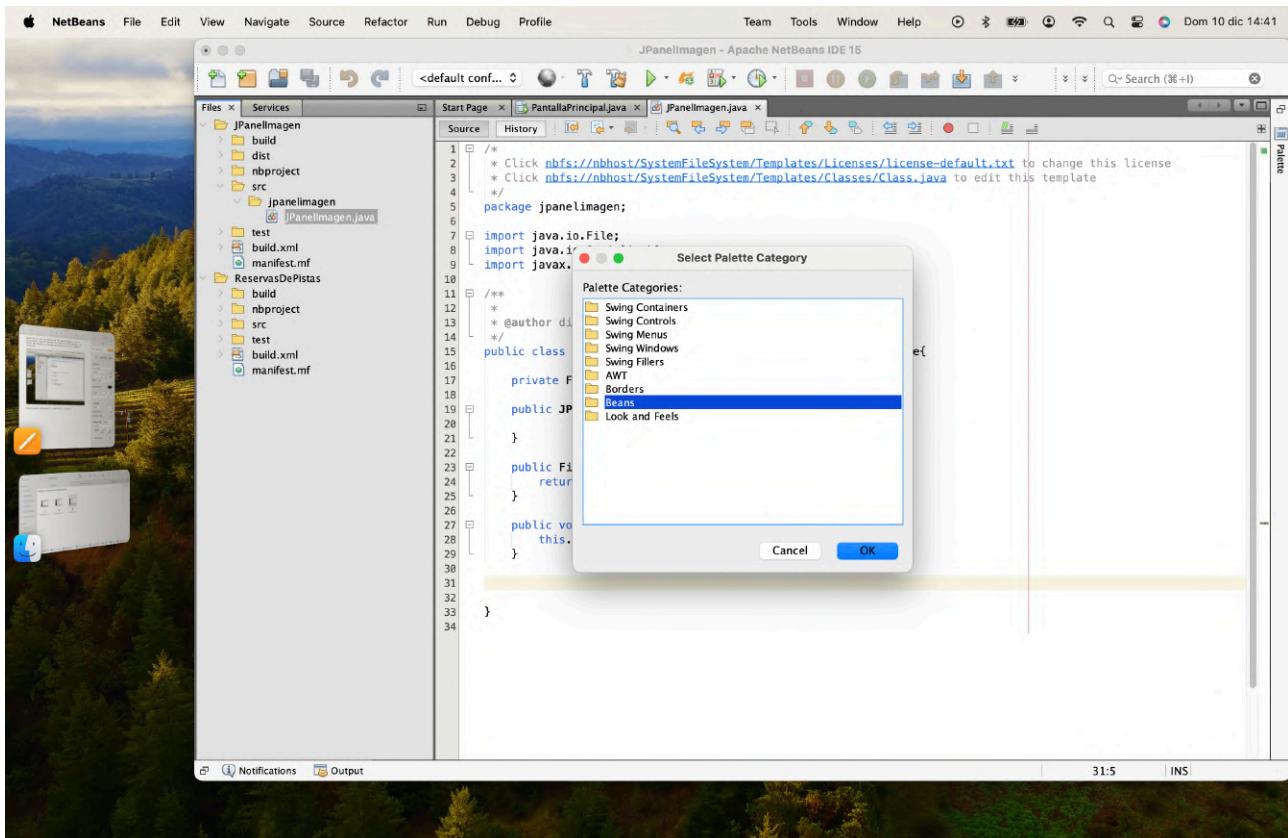
En esta clase también crearemos un método getter and setter para acceder a la ImagenFondo creada en el.

En las siguientes imágenes podemos ver como redefinimos los métodos y creamos el código dentro de dicho método.

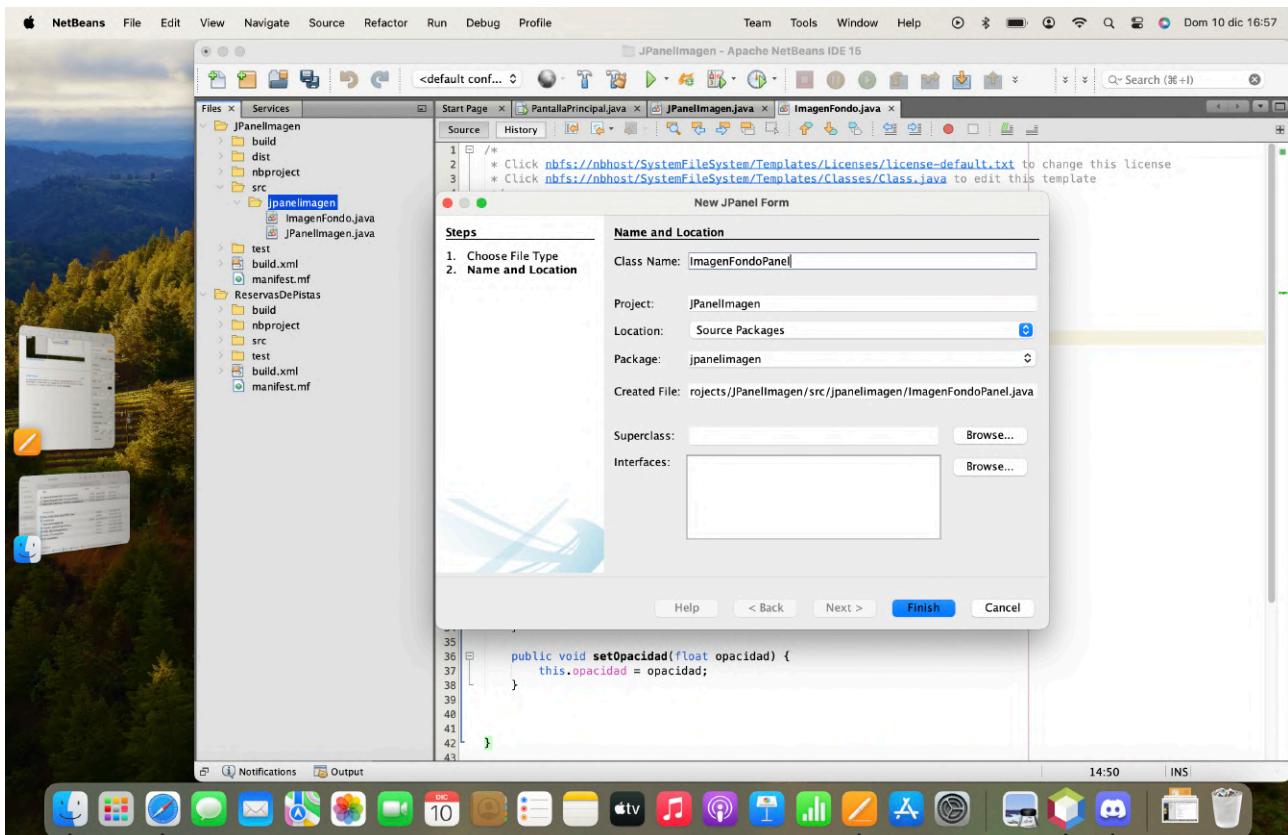




Posteriormente añadiremos el componente a la paleta.

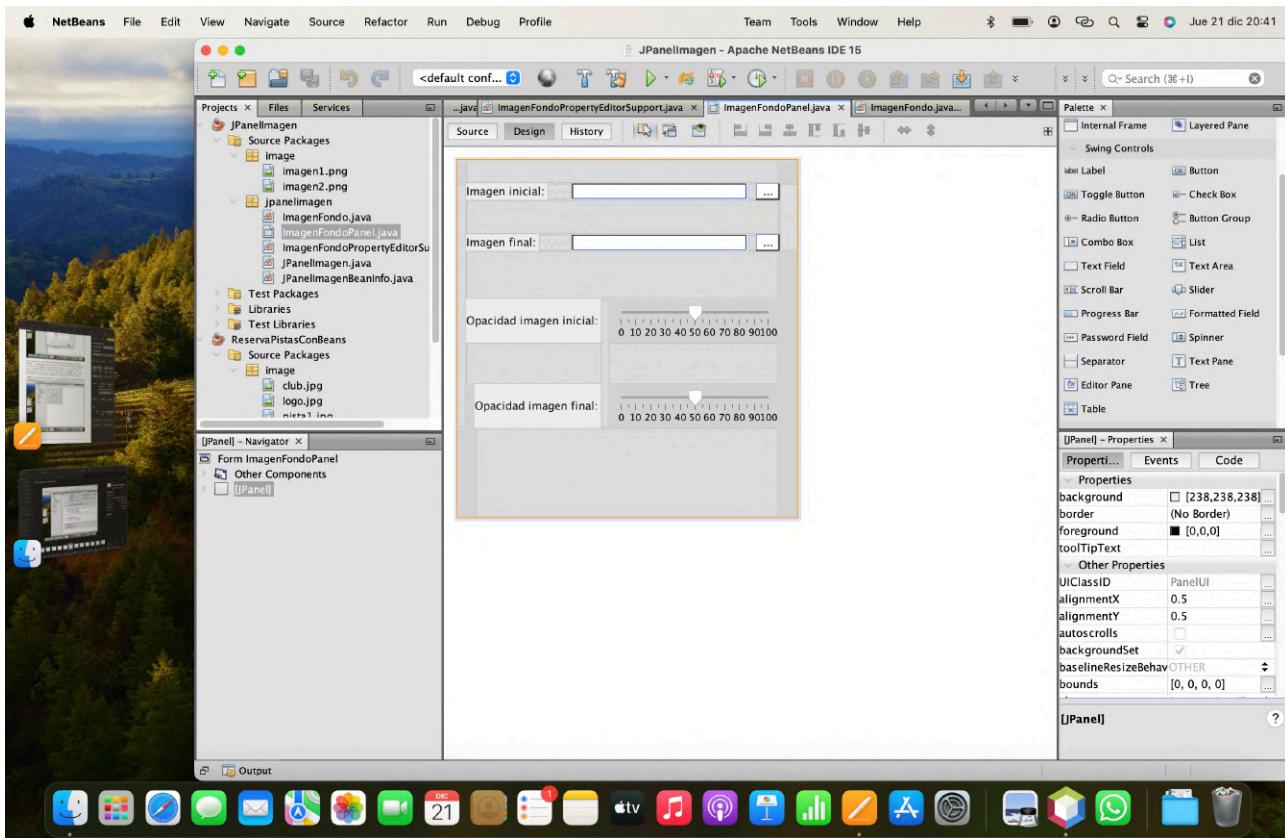


El siguiente paso será crear un JPanel Form llamado ImagenFondoPanel.



El diseño contendrá 2 etiquetas, acompañadas de 1 jTextField y un botón cada una, una para recoger la ruta de cada imagen y otra para buscar cada una de las imágenes. Otras dos etiquetas

acompañaras a dos slider para recoger la opacidad, uno para cada imagen.



En el código del ImagenFondoPanel vamos a crear la función de los botones.

Empezaremos creando una variable privada y estática de la clase ImagenFondo.

Crearemos un JFileChooser llamado fileChooser, una variable de tipo int que capturara si el usuario a seleccionado una imagen y una variable de tipo file llamada file1 para el botón de la primera imagen y file2 para el botón de la segunda imagen.

Mediante un if, se comprobara si se ha seleccionado la imagen (pasándole al variable resultado y comprobando si es igual al método APROVE_OPTION), y si es así, se instanciara en el file de cada botón, mediante el fileChooser creado anteriormente, la imagen seleccionada. Además en el jTextField correspondiente a cada imagen se grabara la ruta de la imagen seleccionada.

A continuación vamos a crear un método getSelectedValue en el cual crearemos dos objetos de tipo file y dos de tipo float que recogerá la ruta de cada imagen y la opacidad (convirtiéndola en un float entre 0 y 1) y retornara un objeto de tipo ImagenFondo con las dos rutas y las dos opacidades recogidas de los jTextField y slider respectivamente.

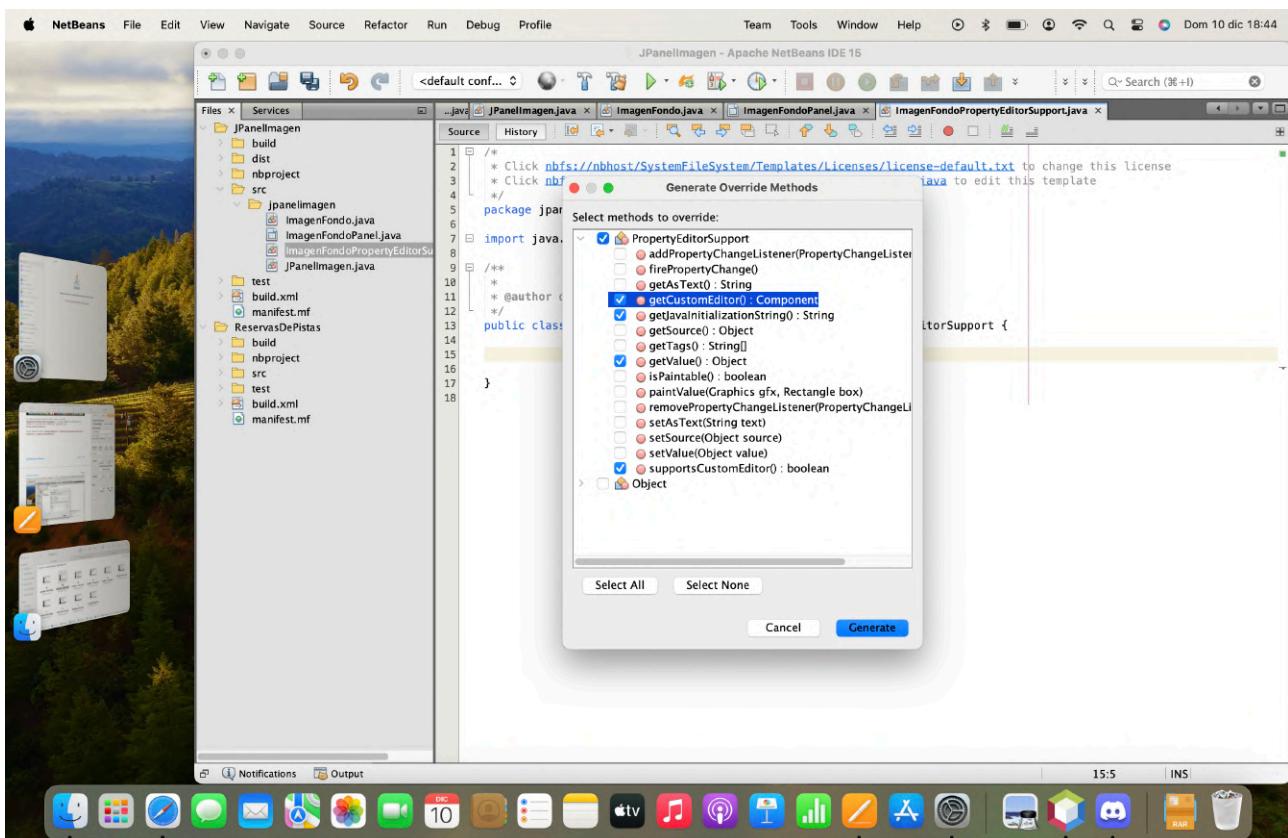
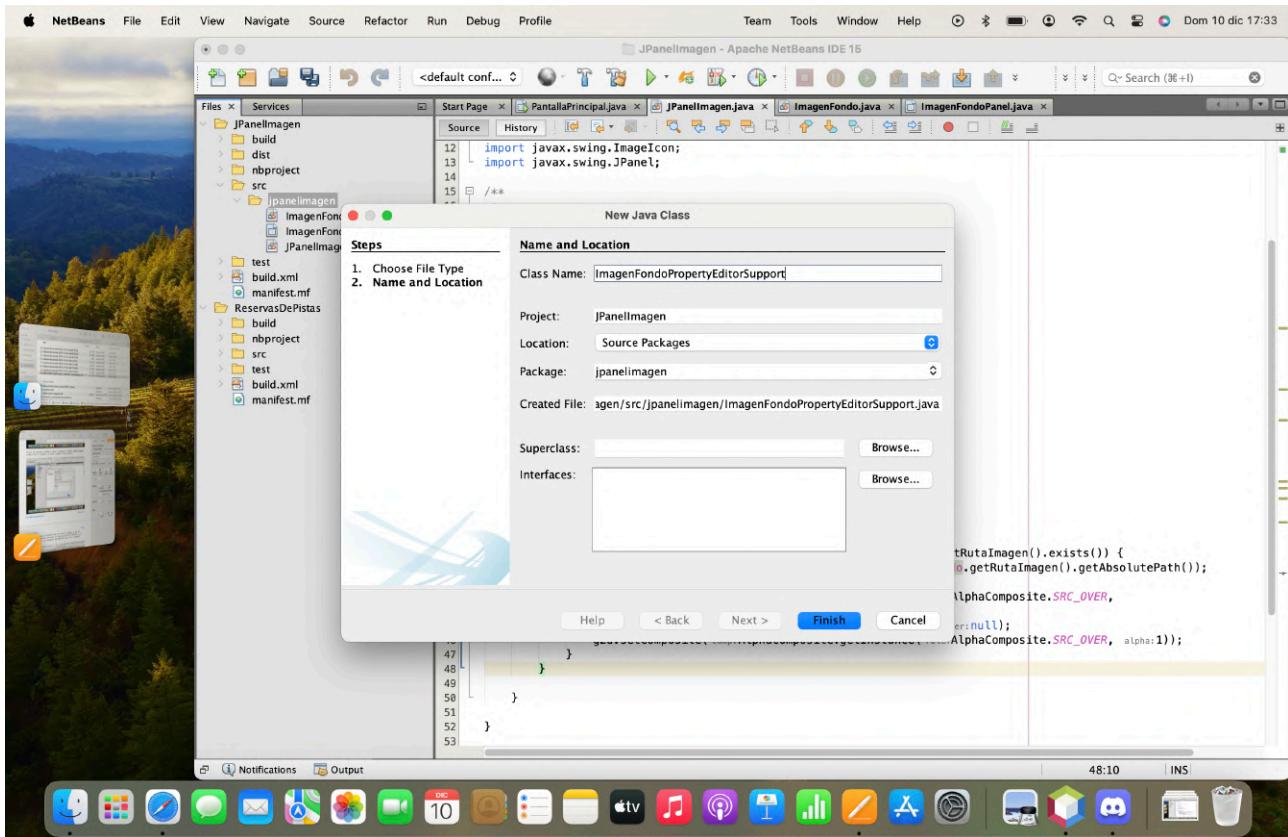
A screenshot of the Apache NetBeans IDE interface. The title bar reads "Apache NetBeans" and "JPanellImagen - Apache NetBeans IDE 15". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a system tray with icons for battery, signal, and date/time. The main window has tabs for Projects, Files, Services, and Java, with "JPanellImagen.java" selected. The left sidebar shows the project structure with "JPanellImagen" as the root, containing "Source Packages" (with "Image" and "jpanelImagen" folders), "Test Packages", "Libraries", "Test Libraries", and "ReservaPistasConbeans" (with "Source Packages" and "Image" folders). The right pane displays the Java code for "JPanellImagen.java", specifically the "ImagenFondoPanel" class. The code handles file selection for image and background files, and includes annotations like @author diego and @SuppressWarnings("unchecked"). The bottom status bar shows "1:1" and "INS Unix (LF)". The Mac OS X Dock at the bottom contains icons for various applications like Mail, Safari, and Finder.

A screenshot of the Apache NetBeans IDE interface. The title bar reads "Apache NetBeans JPanellImagen - Apache NetBeans IDE 15". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a Docked tab for "Jue 21 dic 20:46". The left sidebar shows the project structure under "Projects": "JPanellImagen" contains "Source Packages" (image, jpanellImagen), "Test Packages", "Libraries", "Test Libraries", and "ReservaPistasConBeans" (Source Packages, Image). The "Members" panel below shows class members for "jpanellImagen.ImageFondoPanel". The main editor area displays Java code for "JPanellImagen.java", specifically the "ImagenFondo" class. The code handles file selection, slider values, and variable declarations. The bottom status bar shows "1:1" and "INS Unix (LF)". The Mac OS X dock at the bottom contains icons for various applications like Mail, Safari, and Finder.

Lo siguiente será crear la clase `ImagenFondoPropertyEditorSupport`, la cual tendrá la función de redefinir una serie de métodos y extenderá de `PropertyEditorSupport`.

Estos métodos serán `getCustomEditor`, `getJavaInitializationString`, `getValue` y `supportscustomEditor`.

En las siguientes imágenes podemos ver como creamos la clase y como predefinimos los métodos anteriormente mencionados.



El método supportCustomEditor devolverá true, ya que tenemos un editor personalizado.

El método getCustomEditor devolverá un objeto del tipo ImagenFondoPanel, por lo cual creemos un objeto del tipo ImagenFondoPanel y retornamos dicho objeto.

El método getJavaInitializationString retornara las rutas de las imágenes y la opacidad de las mismas.

Para ello crearemos una ImagenFondo llamada imagenFondo, dos string llamados ruta1 y ruta2 donde guardaremos las rutas de las imágenes y retornaremos un string con las rutas de las dos imágenes y las dos opacidades, todo ello concatenado para que forme un string.

También vamos a sobreescibir getValue que retornara un objeto del tipo ImagenFondoPanel.

En la siguiente imagen podemos ver el código de esta clase.

```

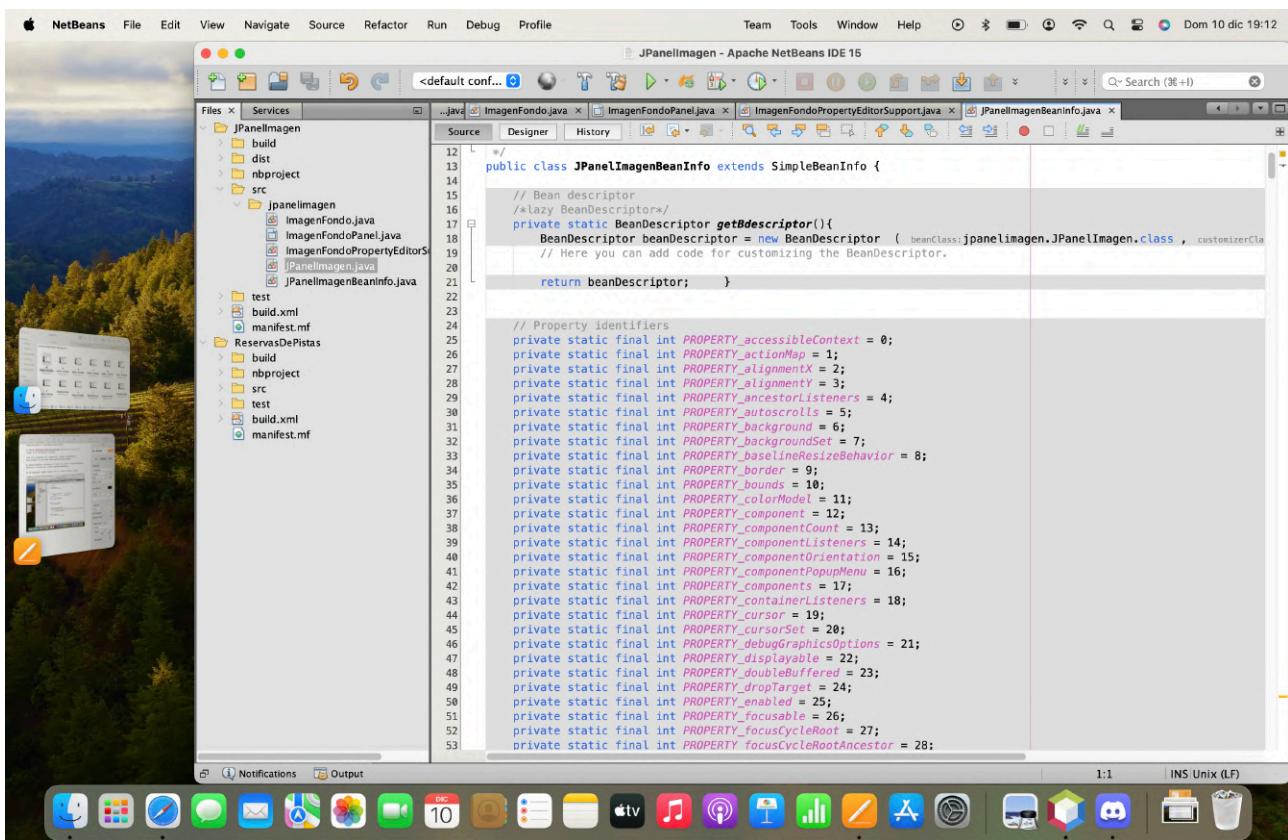
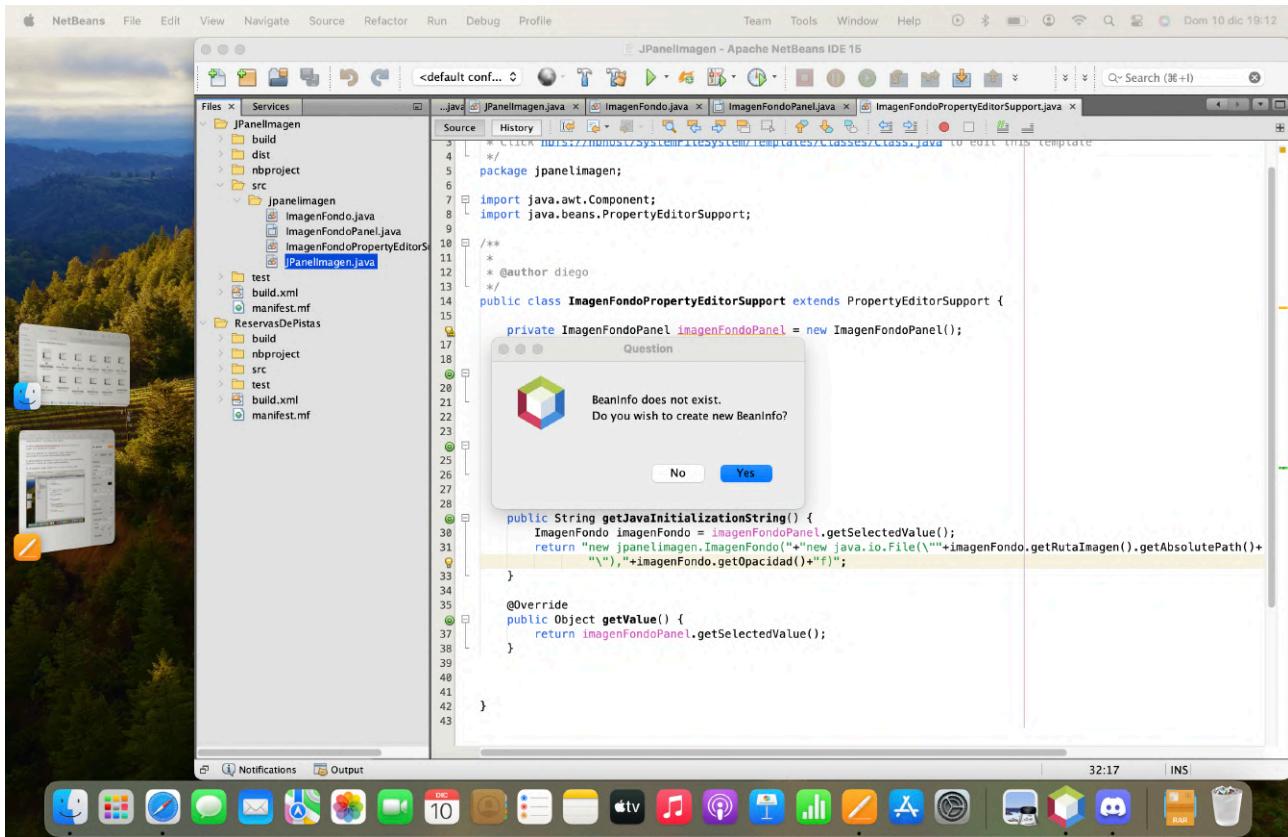
5 package jpanelImagen;
6
7 import java.awt.Component;
8 import java.beans.PropertyEditorSupport;
9
10 /**
11 * @author diego
12 */
13
14 public class ImagenFondoPropertyEditorSupport extends PropertyEditorSupport {
15
16     private ImagenFondoPanel imagenFondoPanel = new ImagenFondoPanel();
17
18     @Override
19     public boolean supportsCustomEditor() {
20         return true;
21     }
22
23     @Override
24     public Component getCustomEditor() {
25         return imagenFondoPanel;
26     }
27
28     @Override
29     public String getJavaInitializationString() {
30         ImagenFondo imagenFondo = imagenFondoPanel.getSelectedValue();
31         String ruta1 = imagenFondo.getRutaImgInicial().getAbsolutePath();
32         String ruta2 = imagenFondo.getRutaImgFinal().getAbsolutePath();
33         return "new jpaneImagen.ImagenFondo(\""+new java.io.File(\""++ruta1+"\"),"+
34             imagenFondo.getOpalInicial()+"f, "+new java.io.File(\""++ruta2+"\"),"+
35             imagenFondo.getOpalFinal()+"f\"";
36     }
37
38     @Override
39     public Object getValue() {
40         return imagenFondoPanel.getSelectedValue();
41     }
42
43 }
44
45 }
46

```

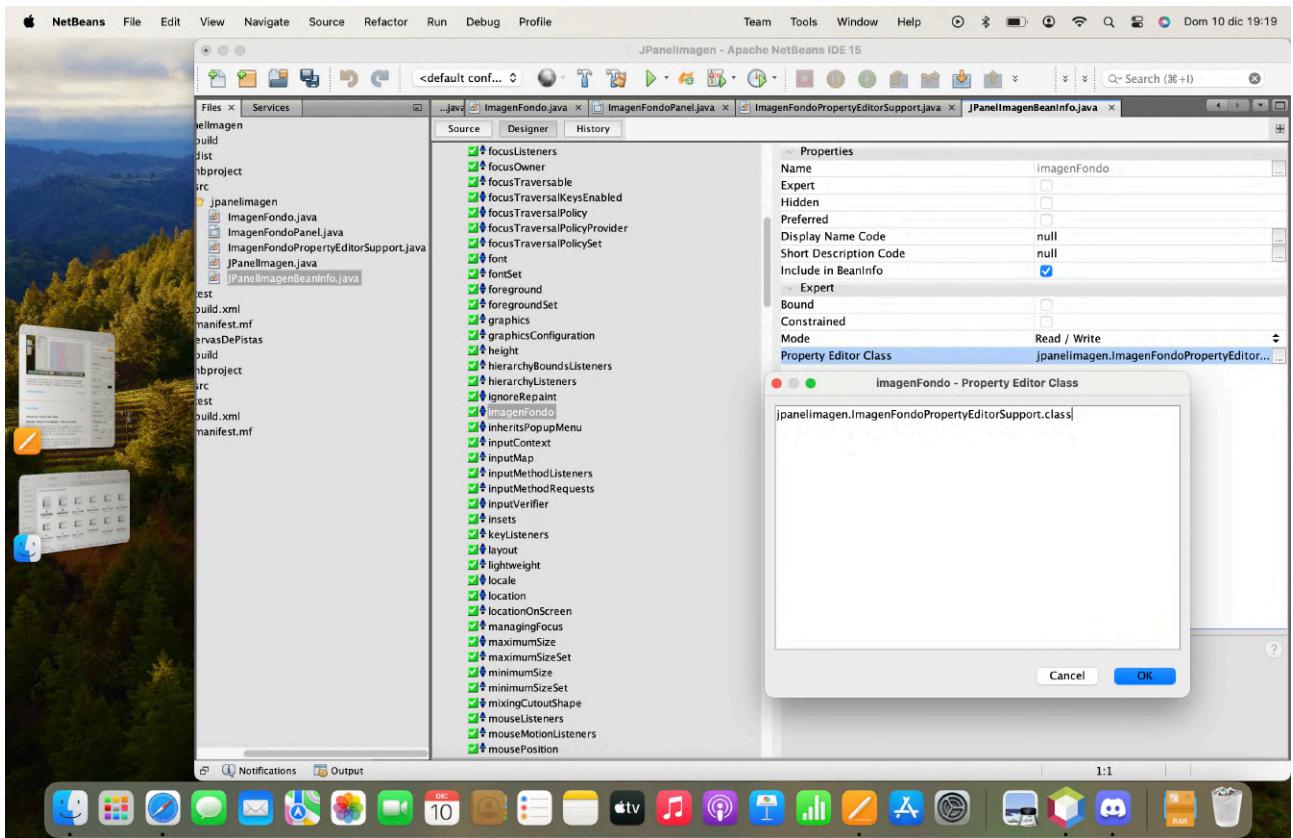
Lo siguiente que haremos será crear el BeanInfo.

Para ello haremos clic con el botón derecho sobre la clase JPanelImagen, posteriormente sobre editor BeanInfo, nos dirá que no existe el BeanInfo y nos preguntará si queremos crearlo a lo cual le diremos que si. Esto nos creara un código en una clase nueva.

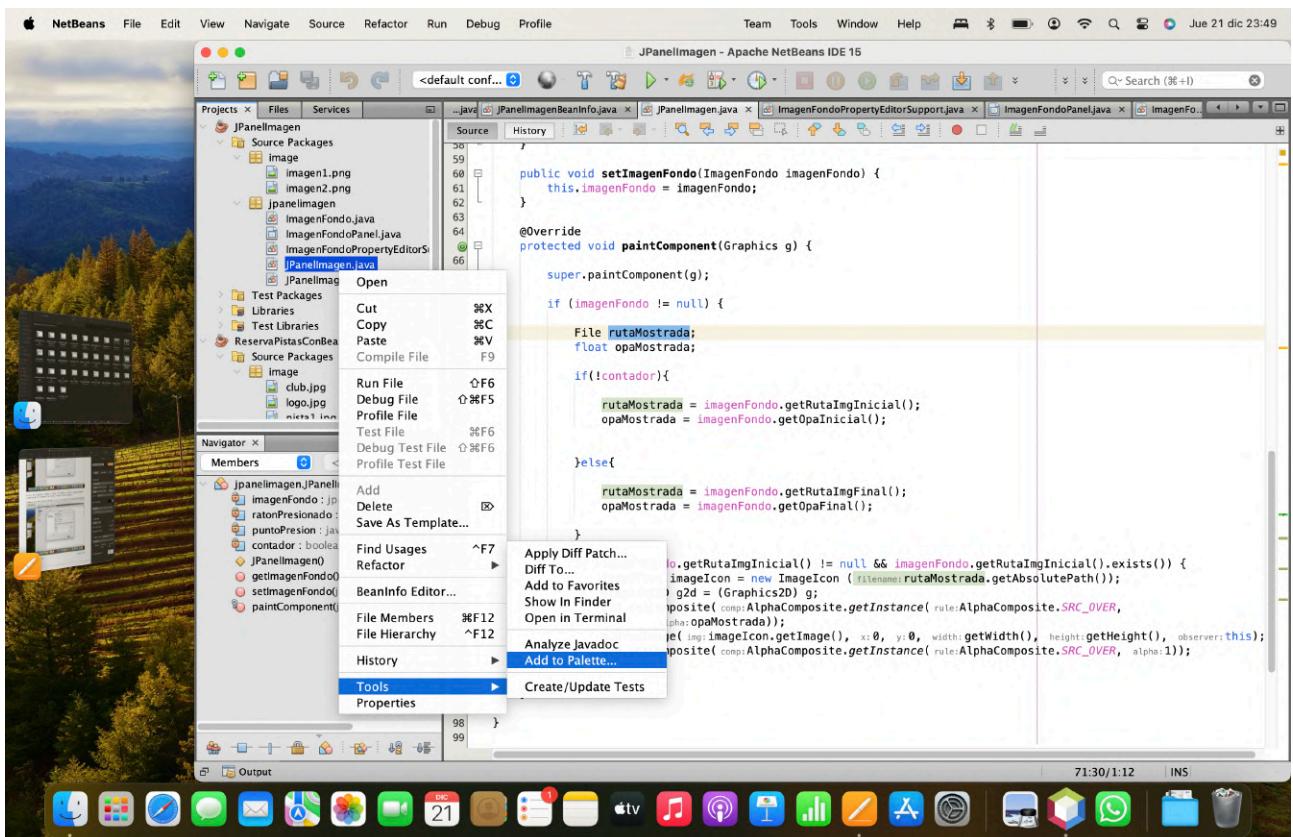
En la siguiente imagen podemos ver como se crea.



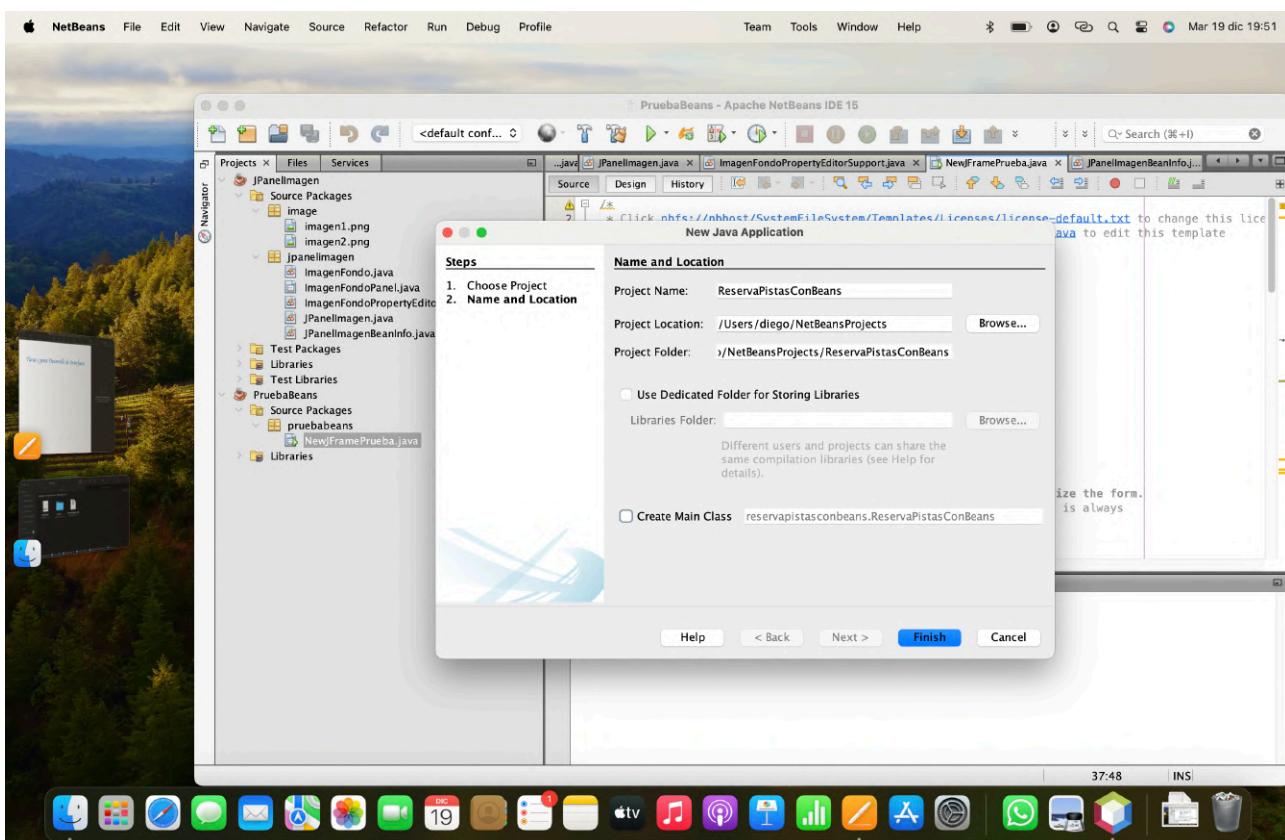
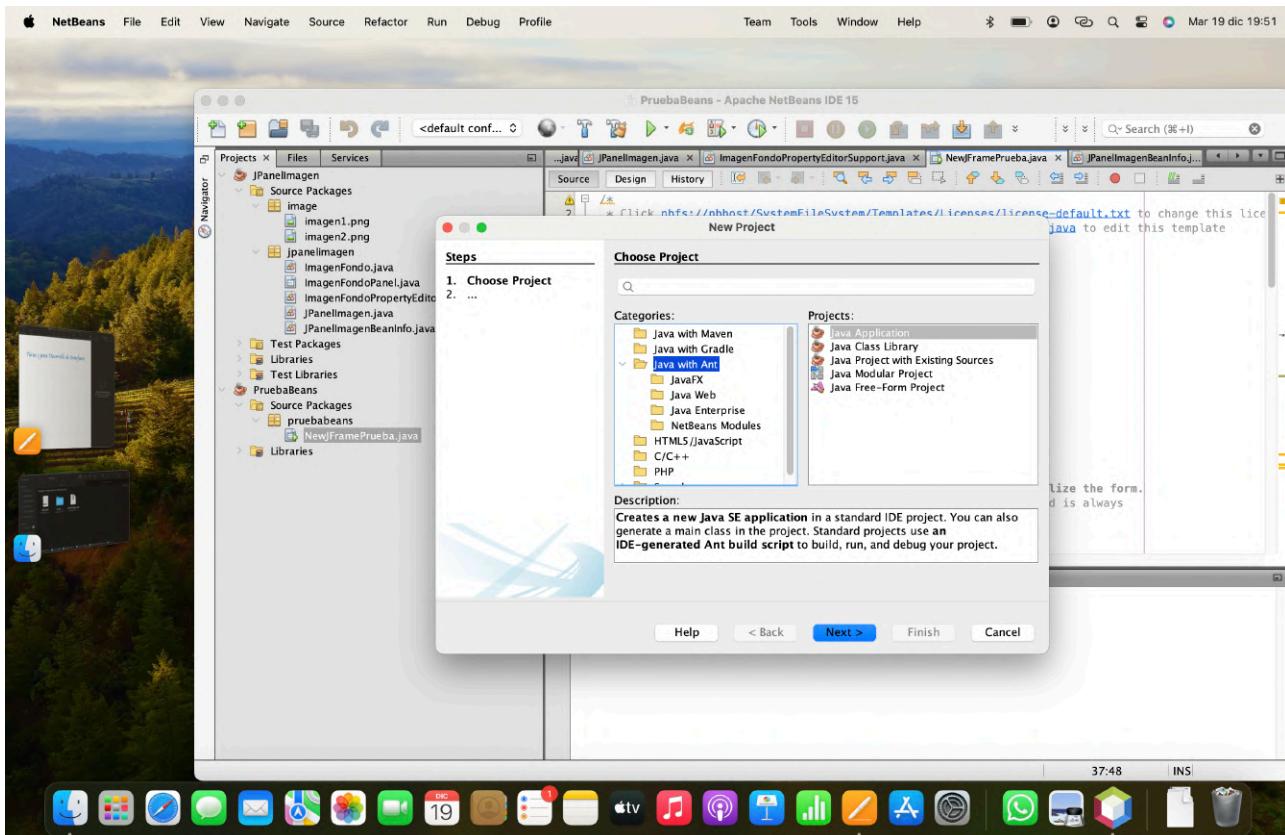
Lo siguiente que faremos será buscar en el designer la propiedad ImagenFondo y le diremos que la clase del editor de propiedades será la que tenemos creada, con paquete incluido, acabada en .class, tal y como puede apreciarse en la siguiente imagen.



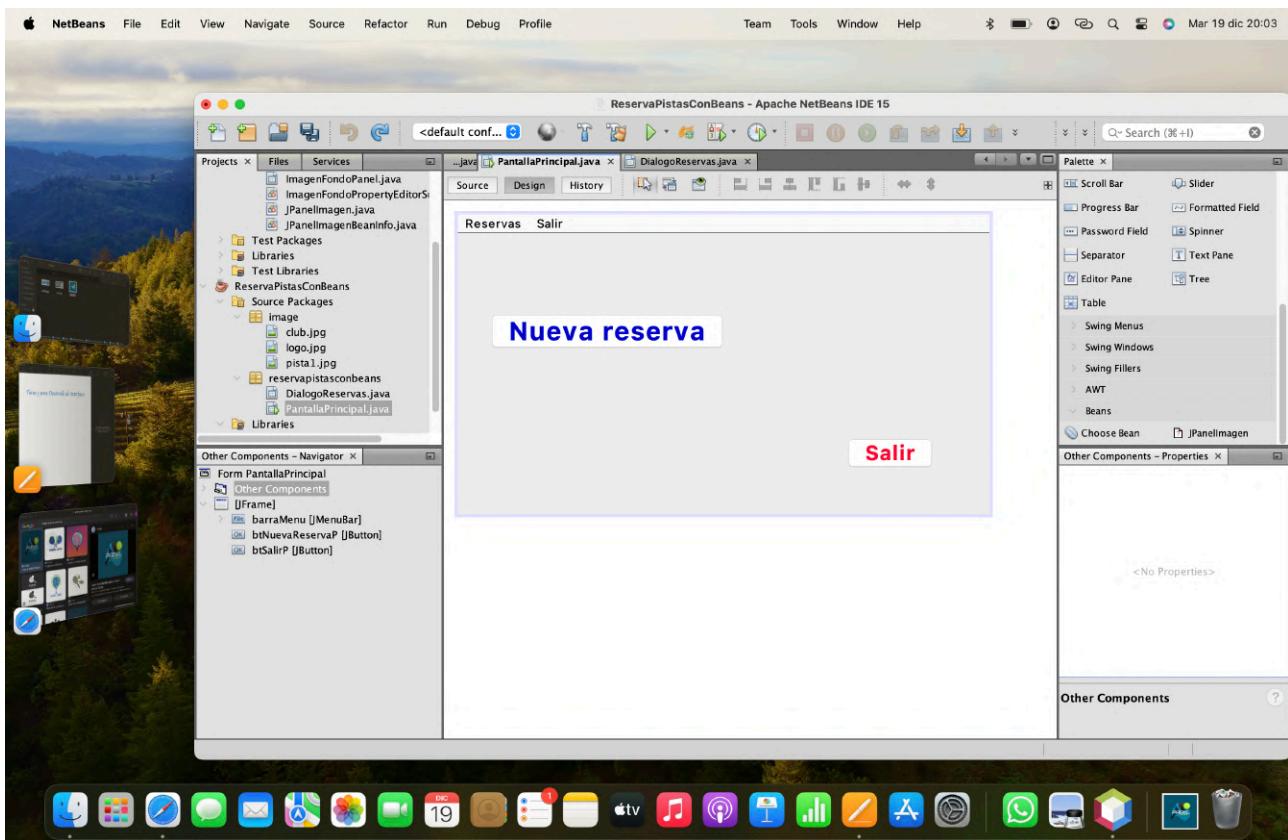
Con esto ya hemos terminado todas las clases por lo que vamos a proceder a añadir a la paleta nuestro jeans.



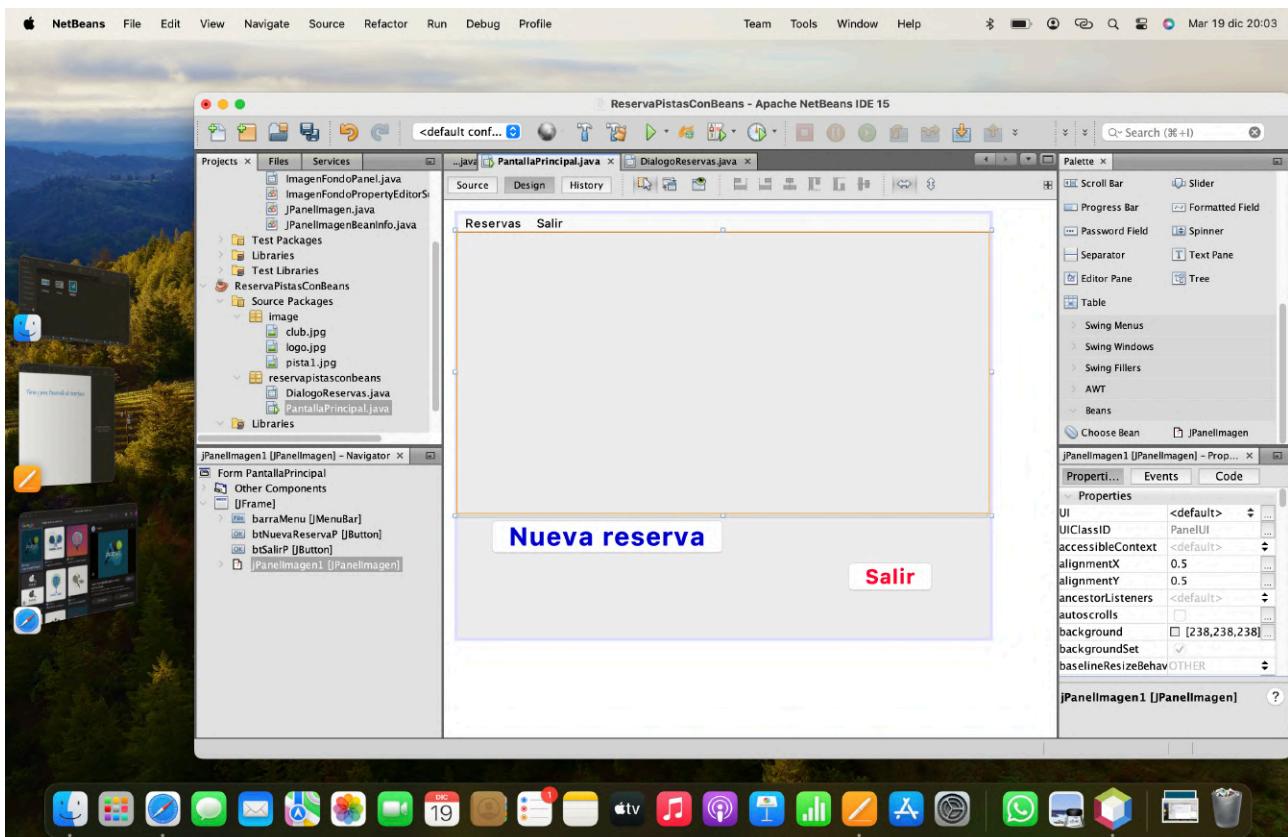
Ahora que ya hemos terminado con el beans vamos a crear un proyecto nuevo llamado ReservasPistasConBeans.



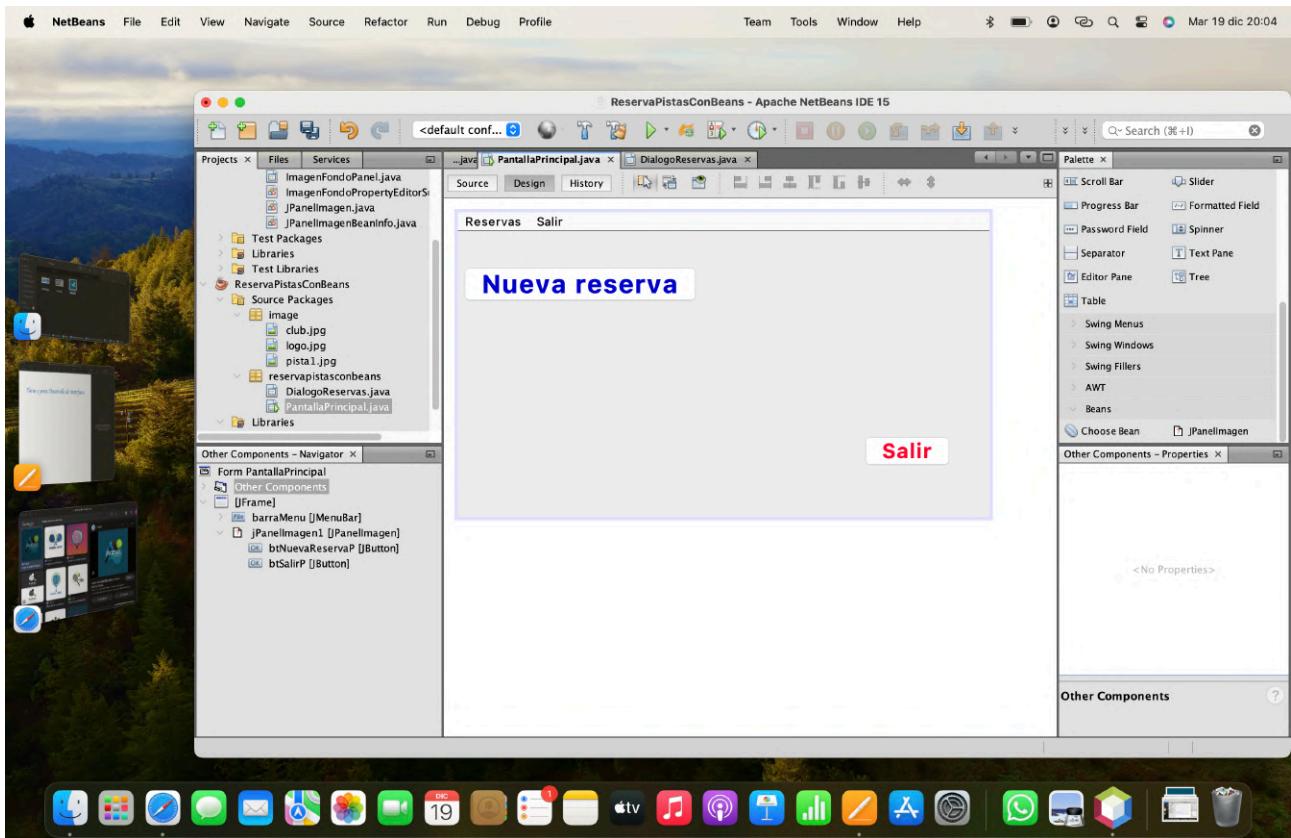
Una vez creado el proyecto vamos a copiar las clases del proyecto creado en la unidad anterior para trabajar sobre él.



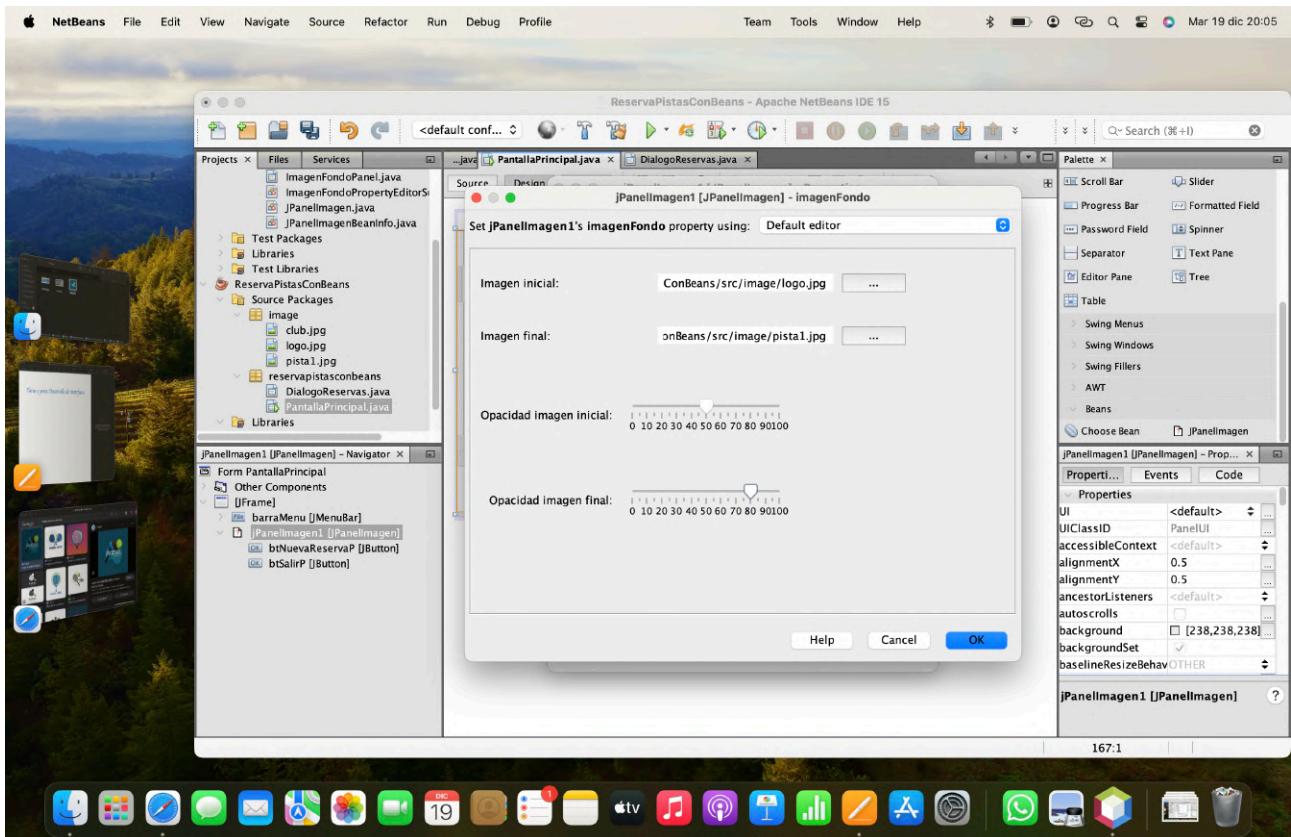
Tras ello vamos a añadir el beans al PantallaPrincipal.



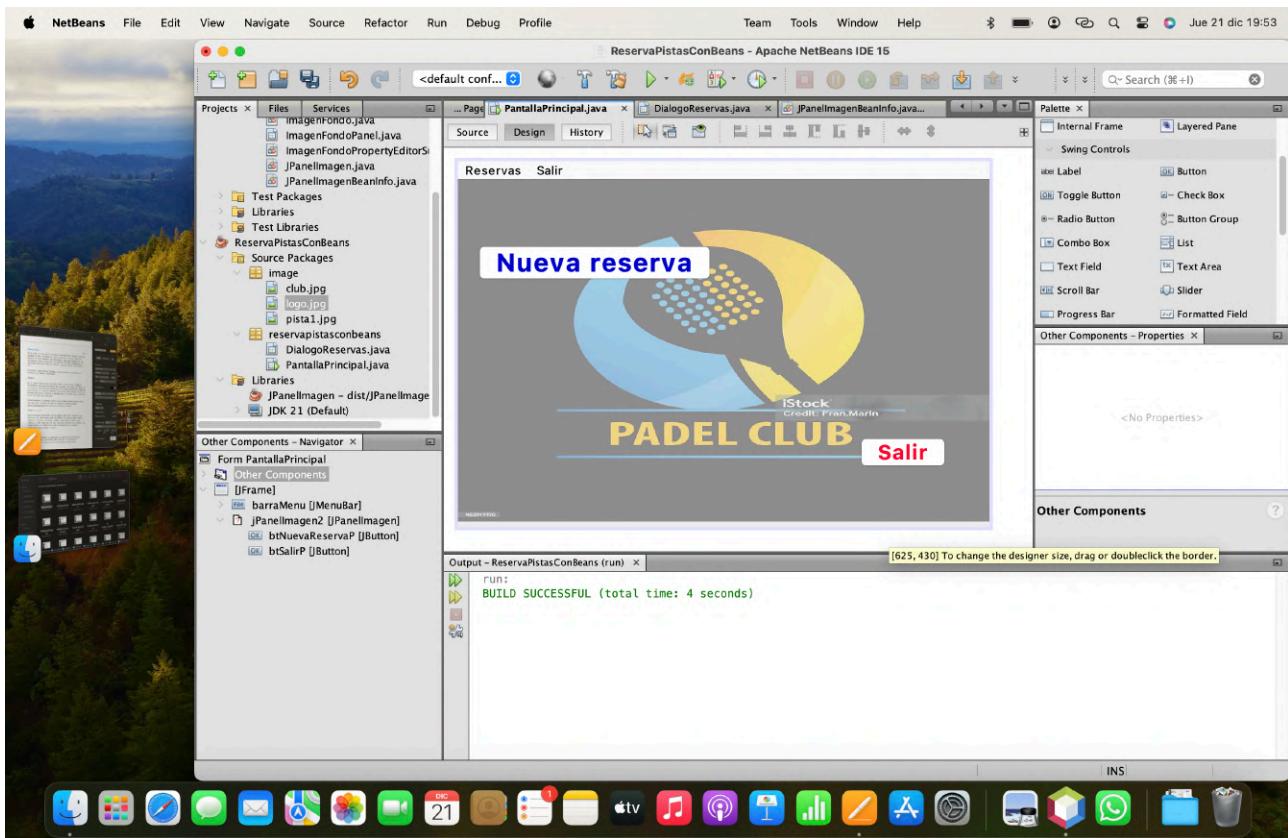
Como podemos observar, al añadir el beans se han movido los botones, por lo que vamos a reubicarlos dentro del beans.



Ahora iremos a las propiedades del beans añadido, buscaremos la propiedad `ImagenFondo` y al seleccionarla nos abrirá el panel que habíamos creado, clicaremos en los botones de buscar las imágenes, las seleccionaremos, daremos la opacidad que queramos y pincharemos en botón de OK.

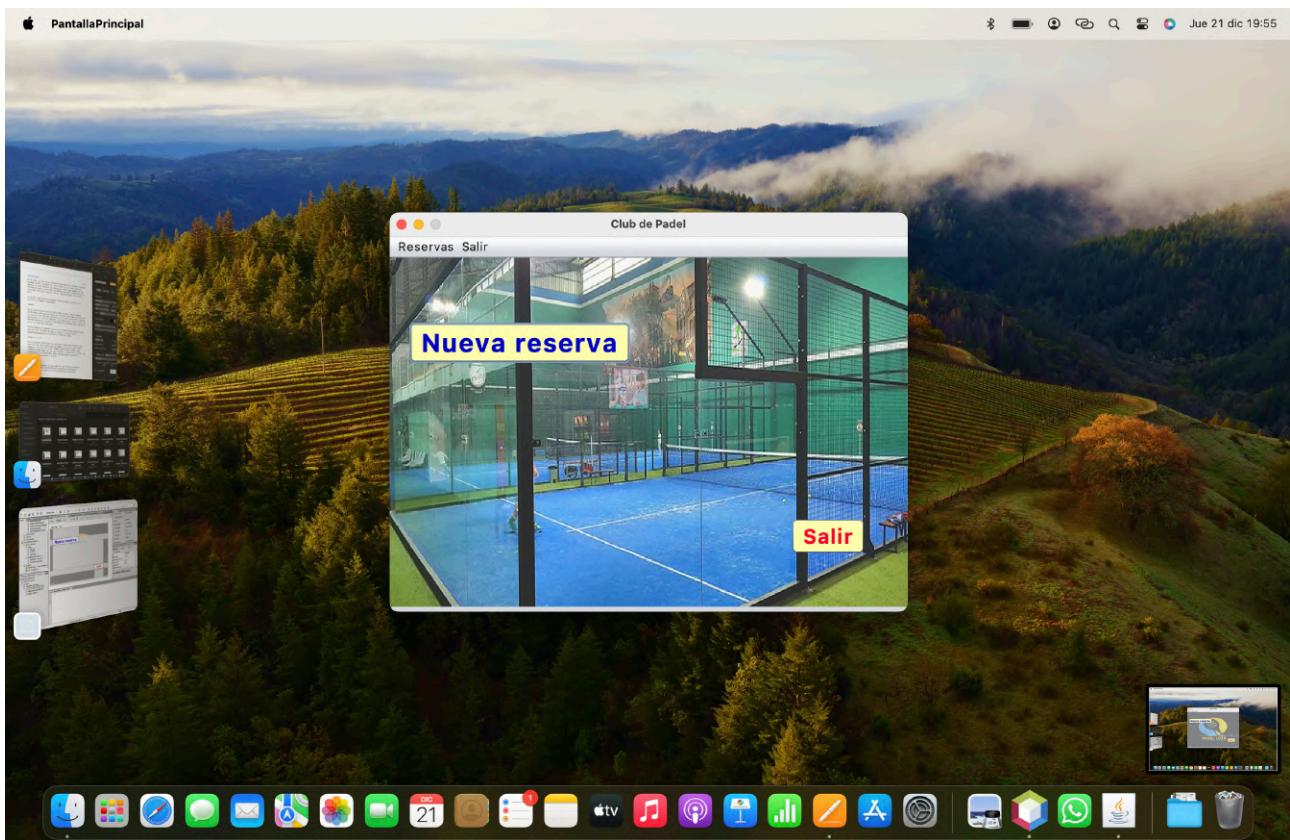


En la siguiente imagen podemos ver como al dar al OK en la ventana anterior ya nos dibuja la primera imagen.



En las siguientes imágenes podemos ver la aplicación funcionando y como ha cambiado la imagen de fondo.





Por último decir que las imágenes no guardan linea temporal ya que he tenido que modificar el proyecto en varias ocasiones.

Y con esto hemos finalizado la tarea.

Detalles de la tarea de esta unidad.

Enunciado. Creación de un componente. Código reutilizable.

Debes crear el componente reutilizable (javabean) que se explica paso a paso en los seis vídeos de la unidad, e incluirlo en la aplicación que desarrollaste para la tarea 1. Este componente será un JPanel personalizado, con las siguientes propiedades:

- Dos imágenes de fondo, una con el icono del club de pádel y otra a tu libre elección (datos de contacto, foto de las instalaciones, etc.), las cuales irán cambiando según se arrastre el ratón verticalmente.
- La opacidad de dichas imágenes.
- Otra propiedad editable de las imágenes del panel (opcional)

Una vez tengas el componente desarrollado, debes utilizarlo en el JFrame y en el JDialog de tu aplicación de reservas de pádel, y mostrar su funcionamiento.

Para ello, será necesario crear, tal y como se explica en los vídeos:

- Una clase cuyos atributos sean las propiedades a definir, es decir, las rutas de las imágenes que se pondrán de fondo y la opacidad que se aplicará sobre ellas.
- Un panel para editar las propiedades de la clase anterior, de manera que permita definir qué imágenes poner en el fondo, junto con su opacidad. Este panel será accesible desde NetBeans cuando se emplee el componente en la aplicación.
- Otro panel que contendrá la primera clase, aplicará la imagen sobre el fondo y será el componente (bean) que se añada a la paleta para pintarlo en la aplicación de ejemplo.
- Una clase que extiende de PropertyEditorSupport para enlazar el panel con el editor de propiedades.
- Un beanInfo que hará referencia a la clase anterior.
- La inserción del nuevo componente (bean) en las ventanas de tu aplicación de la tarea 1 .

La tarea está basada en lo explicado en los vídeos, por lo que síguelos con atención para poder desarrollar el componente. La gestión del evento de arrastre también está explicado al detalle, y **debes cambiar el arrastre horizontal por el vertical** aquí solicitado.

Criterios de puntuación. Total 10 puntos.

- Creación del nuevo componente (bean). 1 punto.
- Creación del panel editor de propiedades que permita definir los valores para imagen1, imagen2, opacidad y otra propiedad modificable de la imagen (esto último es opcional). 1 punto.
- Creación del propertyEditorSupport y el BeanInfo. 1 punto.
- Implementación del getJavaInitializationString, que define lo que se va a incluir en el JFrame cuando se utilice el componente. 1 punto.
 - Gestión del evento arrastrar. 1 punto.
 - Emplear y reutilizar el javabean en tu aplicación. 2 puntos.
 - Generar un fichero con formato pdf con la explicación del desarrollo de la tarea. 3 puntos.

Recursos necesarios para realizar la Tarea.

No necesitas recursos adicionales para realizar esta tarea.

Consejos y recomendaciones.

Visualizar e ir desarrollando la tarea según se indica en los vídeos, adaptando el componente a lo aquí solicitado.

Es aconsejable crear dos proyectos, uno donde se creará el componente y otro donde se (re)utilizará dicho componente.

Indicaciones de entrega.

Una vez realizada la tarea tendrás que comprimir los archivos que has generado y subirlos a la plataforma. La estructura de archivos a entregar dentro del archivo comprimido es como sigue:

- Un directorio llamado **leeme** donde incluyas un archivo .pdf detallando el desarrollo que has seguido para la realización de la tarea (incluye pantallazos).
- Un directorio llamado **proyectos** en el que incluyas:
 - El proyecto Java completo NetBeans en el que hayas creado el componente.
 - El proyecto Java completo NetBeans de prueba y reutilización del componente.

El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_DIx_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna **Begoña Sánchez Mañas para la tercera unidad de DI**, debería nombrar esta tarea como...

sanchez_manas_begona_DI03_Tarea