TEMA 5 – COMPONENTES VISUALES – EJERCICIO 1 - GUIADO

Tener en cuenta que este ejercicio es esencial realizarlo para aprender a definir y usar componentes personalizados y establecer bindings.

- **Creación** y **uso** de un botón personalizado que cambiará su texto y su color de la siguiente manera:
  - Si nunca fue usado, su texto será "Sin enviar" y su color, amarillo.
  - Si se hace clic sobre él por primera vez o después de cancelar: su texto será "Enviado" y su color verde.
  - Si se hace clic después de que se usó para enviar, mostrará el texto "Sin enviar" y el color "rojo".

En la segunda y tercera parte del ejercicio se variará ligeramente el comportamiento del botón a través de SceneBuilder y **binding** respectivamente.
Ejemplo de comportamiento:

"Sin enviar" --> clic --> "Enviado" --> clic --> "Sin enviar" --> clic --> "Enviado"
(Amarillo) (Verde) (Rojo) (Verde)

PRIMERA PARTE DEL EJERCICIO:

La primera parte del ejercicio consiste en la **generación del jar**. Para conseguir el comportamiento, puedes seguir los siguientes pasos:

- Definir una nueva clase que **extienda** de la clase Button.
- Crear tres **propiedades personalizadas**:
  i. "textoSinClic" (**StringProperty**): para almacenar el texto que mostrará el botón en caso de no haber sido pulsado, y en caso de cancelar.
  ii. "textoConClic" (**StringProperty**): para almacenar el texto que mostrará el botón en caso de haber sido pulsado.
  iii. "clicado" (**BooleanProperty**),
    - que tomará el valor true cuando el botón se haya usado por primera vez o tras cancelar:
    - tomará el valor false cuando el botón no se haya usado o tras enviar.
- Además, se deben definir constructores, getters, setters de la clase y un método llamado "cambiarEstado" que es el encargado de cambiar el texto y el color y que será llamado desde un controlador.

Tras la creación de la clase, compilamos para obtener el jar que importaremos desde SceneBuilder.

SEGUNDA PARTE DEL EJERCICIO:

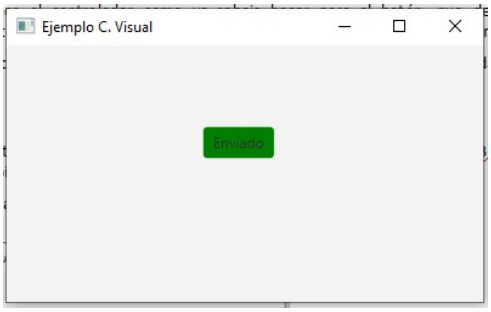
La segunda parte del ejercicio consiste en el **uso** del nuevo componente creado en Java.

- Crearemos una nueva ventana simple en SceneBuilder, importaremos el botón creado y lo añadiremos a la ventana.
- Comprobamos que las propiedades "Custom" del botón aparecen correctamente.
- En java, crearemos el controlador como ya sabéis hacer para el botón, que definirá el método "pulsarBoton". Este método, llamará al método del botón que se encargará de cambiar de estado.
- En SceneBuilder, definiremos el ID del botón, el controlador de la ventana y la llamada a "pulsarBoton" en el onAction del botón.

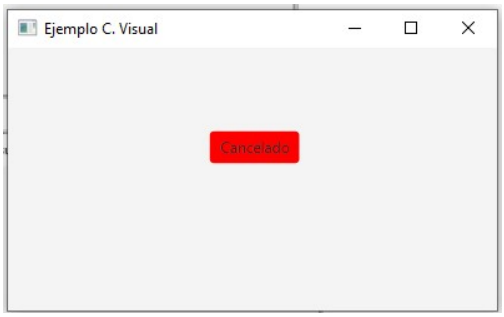
- Una vez que el botón funciona correctamente, comprueba qué ocurre si desde SceneBuilder cambiamos el texto de la propiedad "textoSinClic" a "cancelado".¿Qué conclusión sacas de este cambio?

A continuación, se muestra el ejemplo del comportamiento final del nuevo botón.
Antes de ser usado:

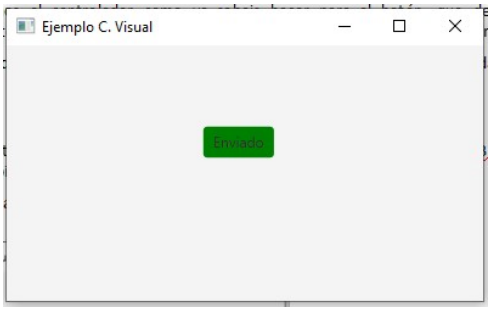
Tras primer clic:



Tras segundo clic:



Tras tercer clic:



TERCERA PARTE DEL EJERCICIO:

- Crear un binding entre dos propiedades del propio botón (modificando la clase Java donde lo personalizamos) para que dicho componente se deshabilite cuando su texto sea “Cancelado”. Para ello debe hacerse uso de:
    - disableProperty()
    - createBooleanBinding()
    - textProperty()

Si necesitas más información puedes consultar todas las propiedades de un botón JavaFX y la documentación a cerca de bindings en los siguientes enlaces:
https://openjfx.io/javadoc/23/javafx.controls/javafx/scene/control/Button.html
https://openjfx.io/javadoc/23/javafx.base/javafx/beans/binding/Bindings.html