Informe TP Final POO

Funcionalidades Agregadas

En cuanto a las funcionalidades agregadas, Decidimos diseñar la clase FigureToggleButton que extiende a ToggleButton, Tomamos esta decisión de diseño porque de esta manera, podíamos hacer que un botón tenga una forma de construir su figura asociada, y esto ofrece la funcionalidad de que al presionar el botón, se pueda dibujar la figura que le corresponde, para que el botón pueda construir su figura asociada, decidimos diseñar la interfaz funcional FigureBuilder, que cuenta con un único método build que devuelve una instancia de Figure, se pide como parámetro en el constructor de FigureToggleButton una instancia de esta interfaz para poder construir la figura correspondiente.

Además, para cumplir con lo pedido por el TP, modelamos las clases Ellipse, Line y Square, para ofrecer la funcionalidad de dibujar estas figuras. También, decidimos modelar la interfaz funcional Movable, con un único método move, para ofrecer la funcionalidad de mover las figuras, hicimos que la clase abstracta Figure y la clase Point implementen esta interfaz.

En cuanto a funcionalidades relacionadas con la UX, hicimos varios cambios. En relación a la creación de figuras, ahora el usuario puede ver en tiempo real la figura que está creando, también si el usuario hace click con el mouse mientras lo está “draguendo”, la operación de creación se cancela, esto lo hicimos para evitar inconsistencias. En cuanto a la selección de figuras, el usuario puede ver el rectángulo de selección múltiple en tiempo real, también al hacer “click” sobre una figura que no está seleccionada esta se selecciona incluso cuando ya hay figuras seleccionadas. Si se hace “click" sobre una figura seleccionada esta se deselecciona. Siempre que haya figuras seleccionadas, estas se muestran en la barra de estado. Otro cambio fue que al tocar cualquiera de los botones que actúan solamente sobre figuras seleccionadas (mover atrás, al frente y borrar) si no hay figuras seleccionadas se indica en la barra de estado que selecciones algunas figuras y se cambia automáticamente al botón de selección. Finalmente al hacer “hover” sobre varias figuras solo se muestra la información de la de arriba en la barra de estado puesto que es es la única seleccionable.

Modificaciones realizadas al proyecto original y cambios a la implementación de la cátedra:

En cuanto a los cambios realizados directamente a la implementación de la cátedra, lo primero que hicimos fue cambiar todas las variables públicas a privadas, de esta manera, nos aseguramos de que estas permanezcan ocultas, con este cambio, tuvimos que agregar métodos para que el frontend pueda interactuar con las variables privadas.

En cuanto a las modificaciones realizadas al proyecto original, modificamos los constructores de las clases de las figuras, esto lo hicimos para que cada figura tenga como comportamiento interno propio su ancho de línea, color del borde y color interno. Además,

también modificamos la clase abstracta Figure, agregando métodos y comportamiento en común para que luego, las clases herederas de Figure, puedan utilizarlo.

Otra decisión de diseño que tomamos, que cambia directamente la implementación brindada por la cátedra, fue modelar la clase ToolBar, esta clase extiende a Vbox y su propósito es brindar mayor modularización. de esta manera, podemos implementar toda la lógica que concierne al funcionamiento de los botones en esta clase y en PaintPane, dejar únicamente código que concierne al dibujado de las figuras y hacer que PaintPane tenga una referencia a una instancia de ToolBar. Con este cambio conseguimos un bajo acoplamiento y una alta cohesión, puesto que antes, los elementos de estos dos módulos estaban mezclados, pero tras esta modificación, los elementos de un mismo módulo permanecen juntos.

Problemas encontrados durante el desarrollo

Lo que más nos causó problemas fue el dibujar las figuras en el canvas, en la implementación original esto se realiza usando instanceof, lo que nos pareció poco ideal. Pensando soluciones para esto, intentamos que cada figura tenga un método que le permita dibujarse a sí misma, para que esto sea posible, las figuras deberían utilizar métodos propios de javaFx. Esta alternativa mezcla el backend con el frontend por lo que decidimos no implementarla y dejarlo como estaba pese a que no sea ideal.

Hasta que comprendimos por completo cuando se llamaban a los eventos del mouse teníamos problemas de interferencia entre los mismos, sobre todo a la hora de querer hacer el rectángulo de selección y que no nos tomará el click.

Al guardar los Colores en las Figuras en el back decidimos crear nuestra propia clase Color, por lo que elegimos no utilizar ninguna implementación de otras librerías, para así generalizar el back y que se pueda utilizar con otros frontEnds

A la hora de testear el correcto funcionamiento del programa, nos resultaba difícil visualizar el rectángulo de selección, por lo que decidimos agregar la funcionalidad de que el mismo se dibuje en tiempo real, así el usuario tiene una preview de lo que está seleccionando. Una vez hecho esto, decidimos agregar esta funcionalidad a todas las figuras.

Al crear el Slider, esperabamos que este tuviese algún método de instancia que permitiese “setear” su título, pues estábamos bajo la idea de que ese atributo era propio de una instancia de Slider, por ello tardamos mucho en darnos cuenta que había que utilizar una instancia de Label y luego agregar el Slider a nuestra ToolBar.