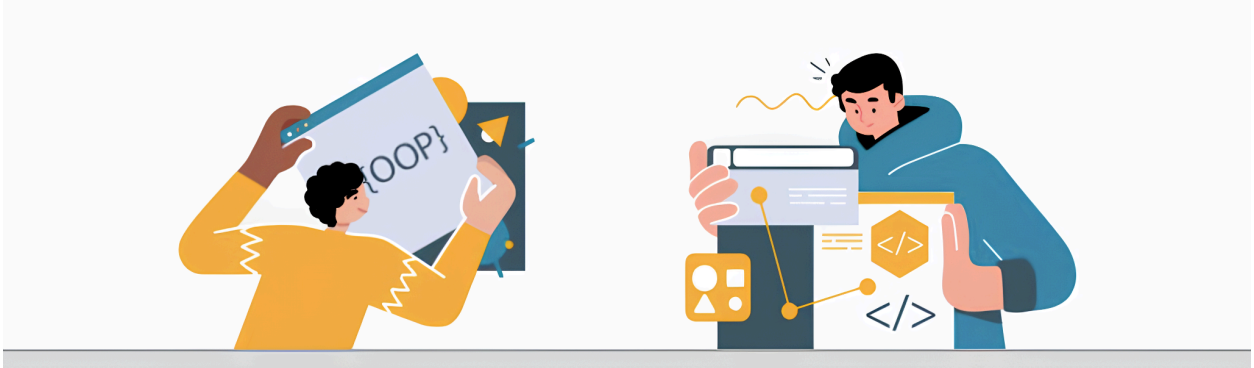
Programación Orientada a Objetos

Final Julio 2024



Grupo 05 - Integrantes:

Lucía Oliveto, 64646, loliveto@itba.edu.ar

Máximo Wehncke, 64018, mwehncke@itba.edu.ar

Tomás Pietravallo, 64288, tpietravallo@itba.edu.ar

Introducción

El trabajo práctico busca modelar una aplicación de dibujo de figuras, implementado en lenguaje Java, con interfaz visual en JavaFX.

Se presenta un lienzo en blanco donde se permite elegir entre varias figuras para dibujar. A cada figura se le puede modificar su color de relleno, su sombra (si es que tiene), su borde, y el grosor de la misma.

Se ofrecen acciones de duplicar, dividir, y mover una figura al centro del lienzo. También se ofrece un sistema de capas, el cual permite dibujar figuras en distintas capas, modificando la visibilidad de cada una.

Cambios hechos a la implementación provista por la cátedra

A continuación se presenta como quedó el proyecto representado por diagramas de UML:

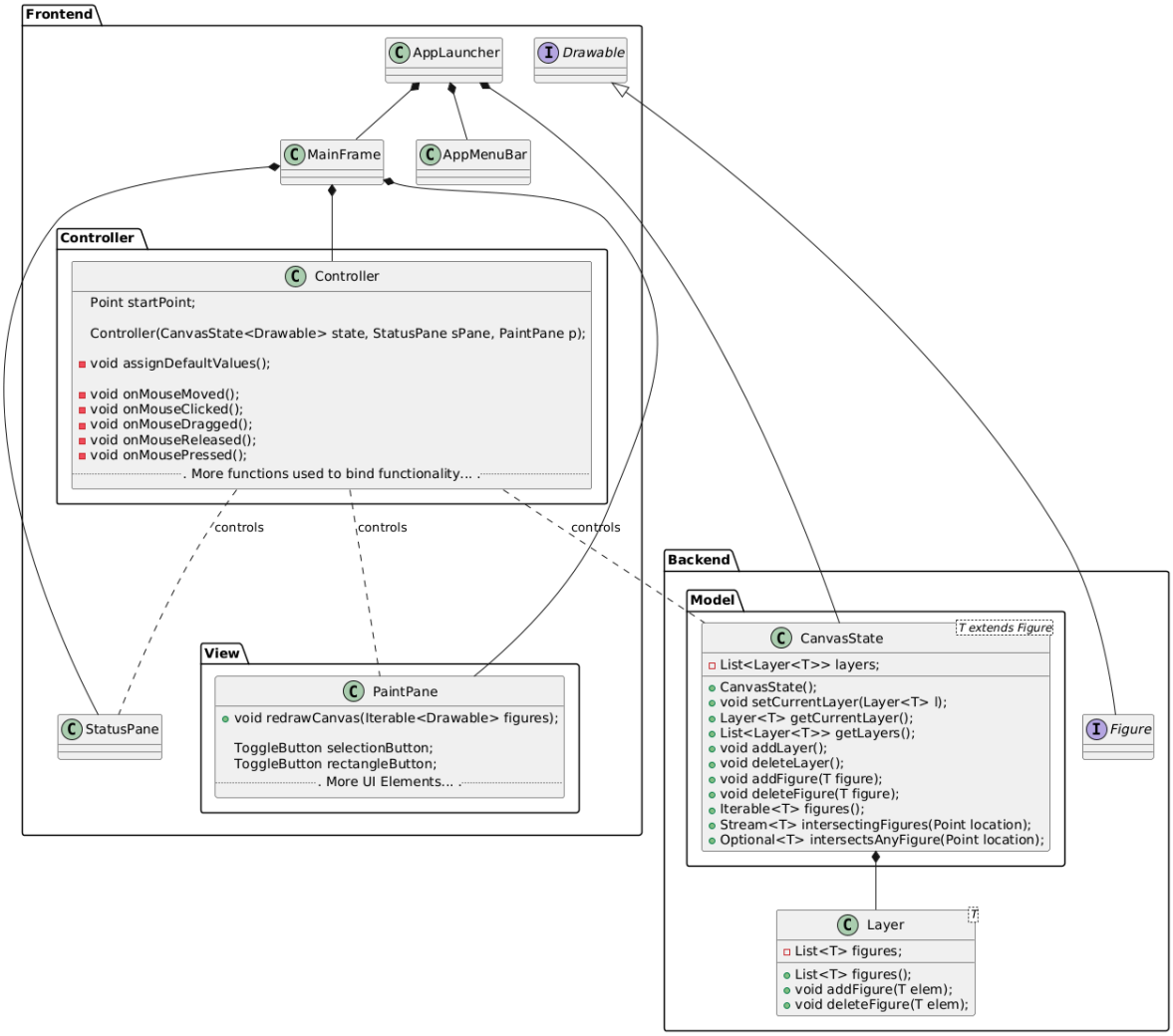


Diagrama que ilustra la estructura Model-View-Controller utilizada en el proyecto.

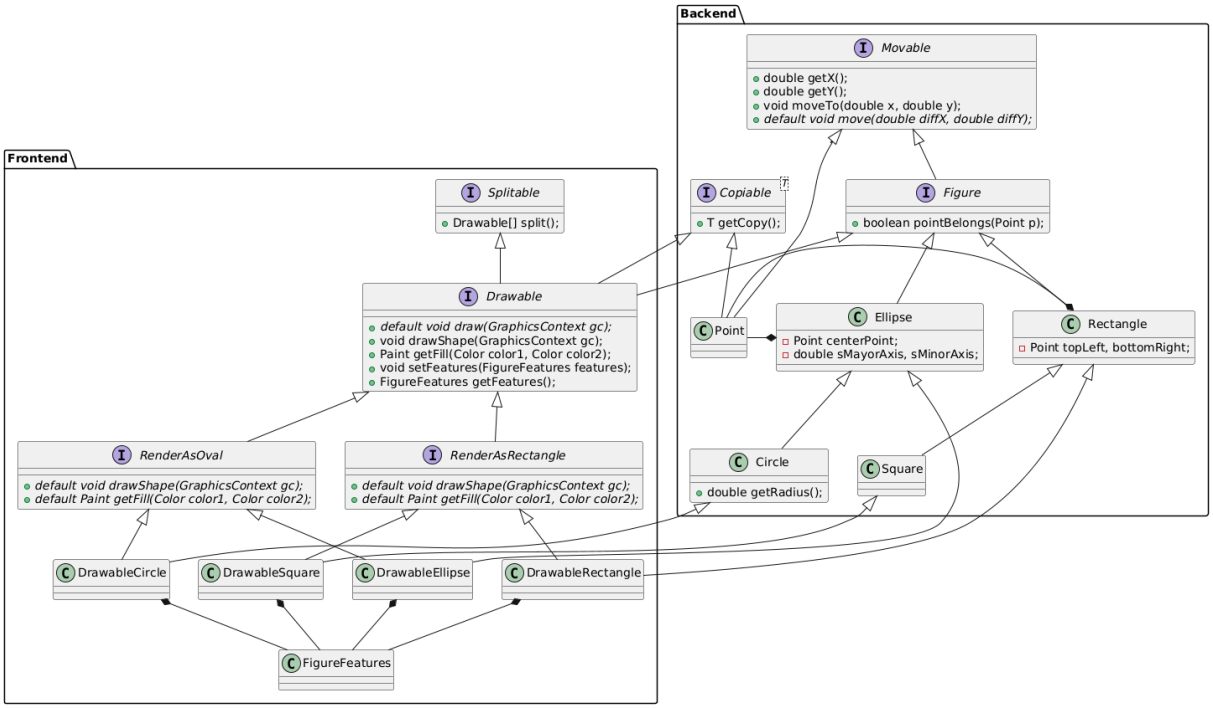


Diagrama UML representando las clases y relaciones para las figuras que se pueden dibujar utilizando la aplicación.

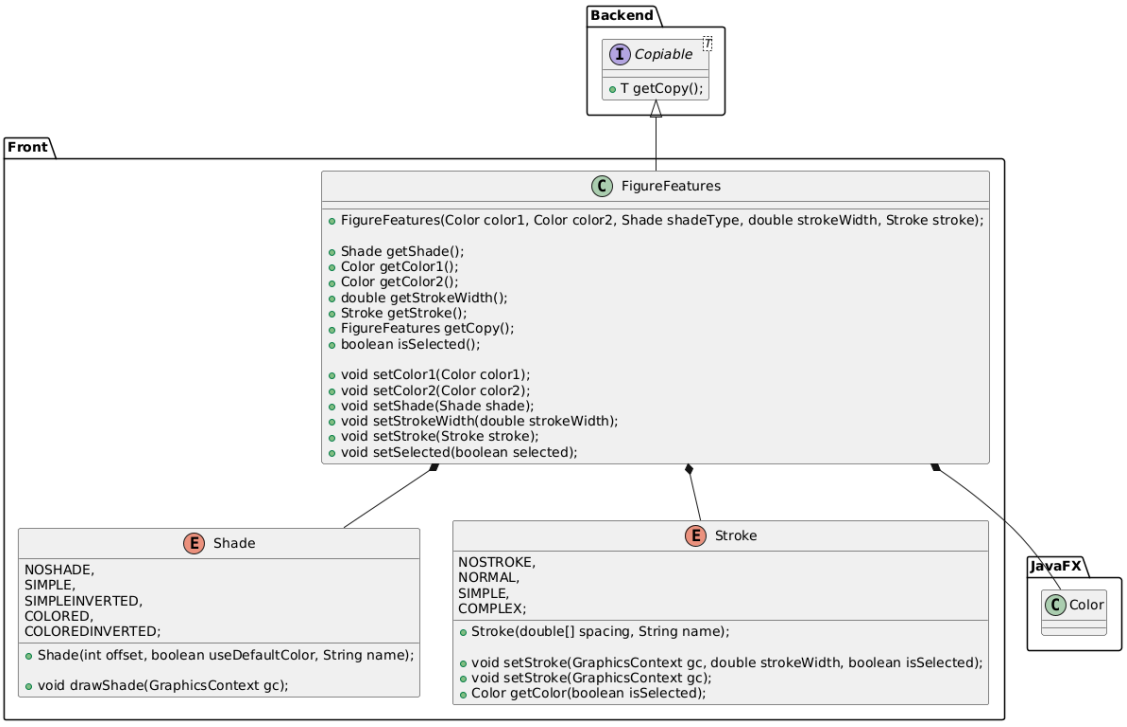


Diagrama UML Representando la relación entre FigureFeatures y Shade, Stroke y JavaFX.scene.paint.Color.

Cambios hechos con respecto al estilo:

- Se modifica la herencia de la clase *Circle*, la cual no heredaba de ninguna clase, haciendo que extienda de *Ellipse.* De igual manera, *Square* extiende a *Rectangle*. De esta forma, se reutilizan funciones ya que la lógica de un cuadrado es similar a la de un rectángulo, y lo mismo con un círculo y una elipse respectivamente.
- Se modifica código de mal estilo de las clases que implementan *Figure.* Como por ejemplo, la visualización de las variables de instancia.
- Se mueve la lógica de las figuras que se encontraba en *PaintPaine* a cada figura, buscando cumplir con el paradigma de la POO.
- Se modifica la visibilidad de variables de instancia en *Point.*
- Se modifican los métodos de *PaintPane* que hacían múltiples *instanceof y se reemplazan por llamados a métodos de instancia de clases que implementan una interfaz común*.
- Se implementa la clase *Controller* que toma la responsabilidad, que tenía antes *PaintPane*, de definir el comportamiento de los elementos visuales y relacionarlos al estado de la aplicación.
- Se modulariza AppMenuBar para no repetir código referido a la creación de alertas al tocar los botones.
- Se movió *selectedFigure* al *back*.

Funcionalidades agregadas

1. Sombra

   Se permite elegir de entre cinco tipos de sombras distintas para una figura. Se implementa un Enum Shade que maneja los tipos de sombras a dibujar.

2. Color

   Se permite cambiar el color de relleno de cada figura, generando un gradiente entre los dos colores.

3. Borde

Se crea un enum *Stroke* con las tres variantes pedidas y una más, llamada *NOSTROKE.*

4. Grosor de Borde

   Se puede modificar el grosor del borde de una figura determinada. Si se lo coloca en cero la figura no tiene borde.

5. Duplicar

   Se toman los datos (tanto puntos como características) de la figura a duplicar y se crea una nueva figura con las mismas características y dimensiones, pero en un punto distinto en el lienzo.

6. Dividir

   Se toman los puntos y las características (color, sombra, etc.) de la figura a dividir para calcular los puntos de las nuevas figuras y asignarles dichas características.

7. Mover al Centro

   Se permite mover una figura al centro del lienzo. Esto se logra moviendo en términos absolutos la figura al centro del canvas (dimensiones máximas sobre dos).

8. Capas

   Se crea la clase *Layer* de forma tal que cada instancia de la misma guarde las figuras que fueron dibujadas en ella. Y en *CanvasState* se guarda una lista con las layers.

Problemas encontrados durante el desarrollo

A continuación, se mencionan las dificultades que encontramos durante el desarrollo:

- Mantener la correcta separación de *front-end* y *back-end.*
- Mantener la consistencia del estado de la aplicación.

- Modularizar, de forma tal que clases a las que se les aplicaba un mismo método y no tenían una interfaz común no implicara múltiples llamadas a una misma función.
- Validación general de condiciones borde del programa.

Aclaraciones

Con respecto a las operaciones *“Duplicar”* y *“Dividir”*, las figuras resultantes de hacer alguna de estas operaciones se crean en la capa actual, no en la capa de la figura “original”. Como no se pueden dibujar figuras en capas ocultas y la creación de nuevas figuras constituye la operación de dibujarlas, si se realiza alguna de las operaciones mencionadas sobre una figura visible estando, actualmente, en una capa oculta, las nuevas figuras no se crearán. En particular, la operación *“Dividir”* eliminará la figura “original” y no creará nuevas figuras.

Link al *issue* correspondiente aquí.