Trabajo Práctico POO
JavaFX - Dibujo de Figuras

Primer cuatrimestre de julio 2024

Grupo 14

Integrantes:

Buela, Mateo	64680
Cuneo Gima, Matias	64492
Prado, Nahuel Ignacio	64276
Lanari, Augusto Andres	64679

Introducción

El siguiente informe detalla lo realizado en el desarrollo del trabajo final de Programación orientada a objetos y los problemas encontrados durante el mismo. La consigna consta de realizar un programa con las funcionalidades provistas por la librería de JavaFX que mostrase figuras 2D en pantalla y se le pudiesen hacer cambios.

Desarrollo

En primer lugar, se alteró el código fuente provisto por la cátedra. Estos cambios fueron:

1. Eliminación de todas las apariciones de instance of
2. Cambio de la clase CanvasState a un ArrayList
3. Cambio a las definiciones de las figuras para que compartan comportamiento
4. Creación de clases figuras tipo frontend
5. Cambio parámetros en el frontend
6. Creación de Enums para botones de creación de figuras
7. Cambios menores de ocultamiento y métodos que se realizaban erróneamente en el frontend

En el primer ítem, se hicieron estos cambios pues la llamada a la función instance of viola el paradigma de la programación orientada a objetos y va en contra de lo visto en la cursada. Para corregir esto, se decidió que aquellos comportamientos realizados de este modo sean total o parcialmente delegados a cada una de las figuras. Este error estaba presente en el método redrawCanvas, en el método figureBelongs, y en el comportamiento seteado a realizarse en el caso de que el usuario arrastre su mouse.
Para el segundo ítem, visto que la clase ya provista contaba con métodos que sólo llamaban a un único método de la clase ArrayList se decidió que, directamente, esta pasase a ser un ArrayList.
Continuando, se realizaron cambios a la estructura de clases correspondiente al backend. Se hizo que las clases Square y Circle heredasen de Rectangle y Ellipse respectivamente, con el fin de que, si compartían comportamiento, ambos no necesiten que se escriba un mismo método.
Se crearon clases en el front end que nacen de la necesidad de que las figuras del back end no pudiesen realizar métodos como el dibujado, entonces estas nuevas contienen la figura de back que representan y sus propiedades que aparecen luego explicadas.
También se cambió los parámetros que se tratan en el frontend de un CanvasState a un mapa de ArrayLists de figuras de front con las Keys sirviendo para la funcionalidad de capas que luego se explica.
Además se diseñó una clase que extiende de ToggleButton que en su constructor recibe el nombre del botón, y un nuevo enum que indica el tipo de botón que se quiere modelar, este último cuenta con los campos `CICLE_BUTTON`, `ELLIPSE_BUTTON`, `RECTANGLE_BUTTON`, `SQUARE_BUTTON` donde cada uno de ellos reescribe el comportamiento de makeNewFigure que retorna una nueva FrontFigure para añadir al lienzo del programa. Esta funcionalidad nos permite eliminar una cadena de if, else if y else que no corresponde con el paradigma.

Finalmente, se cambiaron algunos modificadores de acceso en campos y métodos a lo largo de todo el trabajo pues no eran los correctos según el paradigma, por ejemplo los campos en la clase Point eran públicos cuando debían ser privados y accedidos por getters. Además, habían ciertas operaciones que se realizaban en el frontend que eran más apropiadas en el backend, por ejemplo en el dibujado de las figuras, el método fill requería de la coordenada X del punto más a la izquierda de la figura, la coordenada Y del punto superior y el ancho y largo de la figura. Estos datos se los obtenía mediante operaciones matemáticas y fue cambiado para que cada figura devolviese los mismos de forma que se cumpla lo visto en clase (Tell don't ask).
Luego, se implementaron funciones nuevas al material ya provisto. Estas son:

1. Sombras
2. Relleno con gradiente
3. Tipos y grosor de borde
4. Duplicado de figuras
5. División de figuras en dos partes iguales con la mitad del tamaño original
6. Mover al centro del canvas a una figura
7. División del canvas por capas
8. Etiquetas

La implementación de las primeras tres consta en asociar a cada figura, cuando se crean, los valores que el usuario setee en los toggleButtons y ChoiceBoxes para la sombra, y tipo de grosor, los ColorPicker para los colores y el Slider para el grosor del borde, por medio de una clase Properties. Luego, en redrawCanvas, se usan estos datos para redibujar la figura. Además, si hay una figura seleccionada y se altera uno de los botones antes mencionados, se actualiza el mismo con el valor nuevo.
Para los siguientes tres ítems, se crearon tres instancias de la clase Button que al ser utilizados cuando hay una figura seleccionada se llaman a métodos del backend. Estos realizan las nuevas figuras necesarias en base a la seleccionada con los cambios correspondientes. Y, finalmente, en caso de ser necesario, les asigna las propiedades mencionadas en los primeros tres ítems de la figura original.
Para la división por capas, como antes se menciona, se utiliza un mapa con figuras como llaves y de valores un par Boolean y ArraysLists de figuras del front que cuando se crean nuevas figuras se las añade al mapa según en qué capa se están creando. Luego, en el método redrawCanvas se van dibujando las figuras según en qué layer están y si el valor booleano asociado es verdadero, respetando el orden de la consigna. Para los botones, se diseñó una nueva clase de frontend donde se encuentra tanto el diseño y funcionamiento del hBox, choiceBox, radioButtons y buttons para tanto elegir la capa a ubicar la figura, mostrar, ocultar, agregar y eliminar las capas.
Para las etiquetas modelamos una nueva clase que extienda de borderPane para el panel superior y luego añadimos a PaintPane una caja de texto y un botón de guardado para justamente poder asignar un texto, unas etiquetas a una figura. Para la asignación de etiquetas, figuras añadimos a Properties un HashSet de Strings, junto con los métodos setTags y getTags. SetTags mediante polimorfismo puede recibir o un Set de Strings o un String literal el cual será fragmentado usando el método split y luego almacenado como un HashSet. GetTags devuelve un nuevo HashSet de Strings que luego es utilizado para preguntar si contiene una tag específica. Para preguntar por tags específicas se utiliza GetTags en conjunto con contains y se le pasa un string que pertenece a LabelsPane mediante un getter, lo llamaremos filtro. LabelsPane contiene una caja de texto a la cual le

añadimos un listener para saber cuando cambia y modificar el String filtro para que solo contenga la primera palabra. Finalmente desde redrawCanvas se modifica su comportamiento para que se dibujen solo las figuras seleccionadas por el filtro, respetando las capas.
Por último, no consideramos haber tenido grandes problemas en el transcurso del trabajo. Sin embargo, si hubieron menores inconvenientes, principalmente de incertidumbre con lo pedido en la consigna, que fueron preguntados y contestados en los espacios provistos por la cátedra y, además, como el grueso del trabajo se realizó en solo unos pocos archivos, se decidió, que para evitar problemas con git, que la mayoria del codigo lo escribiese uno a la vez y que el resto de los integrantes lo aconsejen o le dicten. También, como no sabíamos la condición de la aprobación de cursada de uno de los integrantes se decidió que se abriese una nueva branch en el repositorio para realizar la consigna pertinente a los grupos de cuatro integrantes. Finalmente, se decidieron hacer algunos grandes cambios al trabajo poco antes de la entrega y para ello se decidió emplear otra branch más.

Conclusión

Durante el transcurso del trabajo práctico nos fue posible aplicar y expandir los conocimientos adquiridos a lo largo de la materia. Por otro lado, a su vez, se adquirieron nuevos conocimientos, como el uso de JavaFX para interfaces gráficas e interfaces gráficas en general.