2023

Taller 5 – Patrones de Diseño

DISEÑO Y PROGRAMACIÓN ORIENTADA A OBJETOS

DANNA LUCIA CASTILLO PALACIOS

UNIVERSIDAD DE LOS ANDES | Pregrado en Ingeniería de sistemas

Tabla de contenido

1.0 Librería JGraphX

1.1 Información general del proyecto

1.1.2 Propósito y funcionalidad

JGraphX se utiliza para construir y representar visualmente grafos en aplicaciones Java. Proporciona un conjunto completo de funcionalidades para crear, editar y visualizar grafos, así como interactuar con ellos. JGraphX puede utilizarse en una amplia gama de aplicaciones que involucren representaciones gráficas de relaciones y estructuras complejas.

JGraphX admite una amplia gama de funciones para permitir que la visualización de celdas solo esté limitada por la habilidad del desarrollador y la plataforma Swing. Los vértices pueden ser formas, imágenes, dibujos vectoriales, animaciones, prácticamente cualquier operación gráfica disponible en Swing. También puede usar marcado HTML en etiquetas de texto. (JGraph, 2020)

1.1.2 URL del Proyecto

https://github.com/jgraph/jgraphx.git

1.2 Estructura general del diseño

1.2.1 Generalidades del diseño

El diseño de JGraphX sigue una estructura modular y jerárquica. La biblioteca se compone de varios componentes clave, como mxGraph, mxGraphModel, mxCell, mxGeometry, mxStylesheet y mxGraphView, que interactúan entre sí para proporcionar la funcionalidad completa. Estos componentes se basan en principios de diseño orientado a objetos y patrones de diseño para facilitar la extensibilidad y la personalización.

1.2.2 Retos de diseño

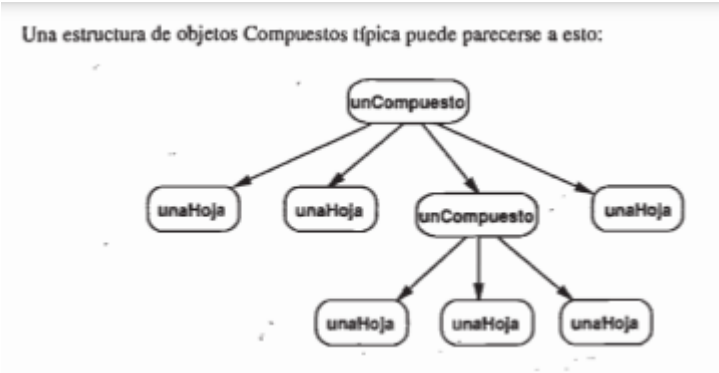
- Representación eficiente de grandes grafos: JGraphX debe ser capaz de manejar eficientemente grafos grandes con miles o incluso millones de elementos.
- Rendimiento y optimización: La representación y manipulación de gráficos pueden ser operaciones intensivas en términos de rendimiento. JGraphX debe optimizar su implementación para proporcionar una experiencia fluida incluso con grandes gráficos y operaciones complejas.
- Coherencia visual y estilos: Mantener la coherencia visual de los elementos del grafo y aplicar estilos consistentes en toda la representación gráfica puede ser un desafío, especialmente cuando se trata de elementos interconectados.

2.0 Patrón Compuesto (Composite)

2.1 El patrón Compuesto

El patrón de diseño Composite es un patrón estructural que permite tratar a los objetos individuales y a las composiciones de objetos de manera uniforme. Proporciona una estructura jerárquica en forma de árbol, donde los nodos hoja y los nodos compuestos se tratan de la misma manera.

En el patrón Composite, se define una interfaz común llamada "Componente" que representa tanto a los objetos individuales como a las composiciones de objetos. Los objetos individuales se denominan "hojas" y los objetos compuestos se denominan "compuestos". Los compuestos contienen una lista de componentes hijos, que pueden ser tanto hojas como otros compuestos.



2.2 Usos frecuentes del patrón Compuesto

El patrón Composite se utiliza cuando se desea tratar los objetos individuales y las estructuras de objetos compuestas de manera uniforme. Algunos casos de uso comunes del patrón Composite son:

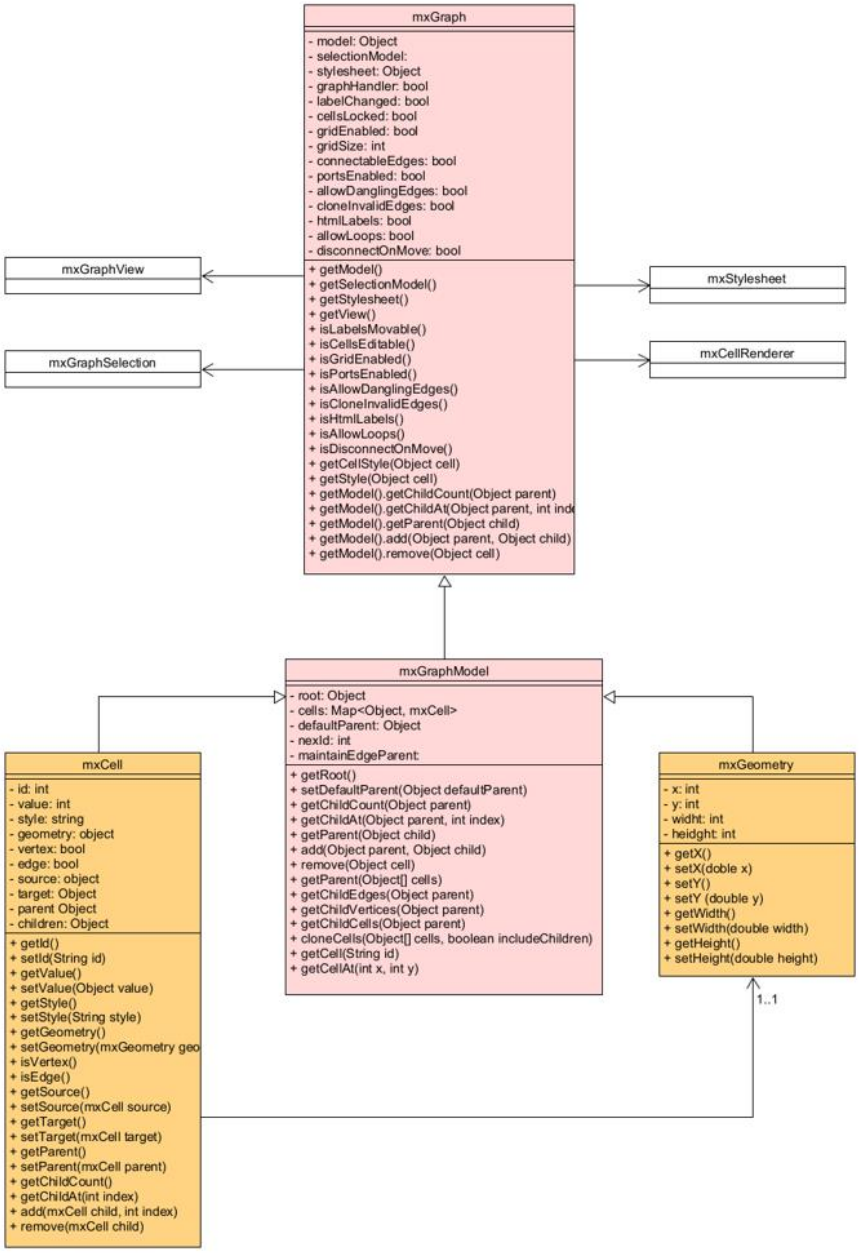
- Representación de estructuras jerárquicas: El patrón Composite permite representar estructuras jerárquicas complejas mediante la composición recursiva de objetos.
- Tratamiento uniforme de elementos individuales y compuestos: El patrón Composite permite tratar tanto los elementos individuales como los compuestos de la misma manera, lo que simplifica el código y mejora la flexibilidad.
- Recursividad y composición de comportamiento: El patrón Composite permite aplicar operaciones recursivas en la estructura jerárquica de objetos, como realizar un cálculo o aplicar una operación a todos los elementos.

- Manipulación eficiente de estructuras complejas: El patrón Composite facilita la manipulación de estructuras complejas mediante el acceso y la modificación uniforme de los elementos individuales y las composiciones de elementos.

3.0 Implementación del Patrón compuesto en JGraphX

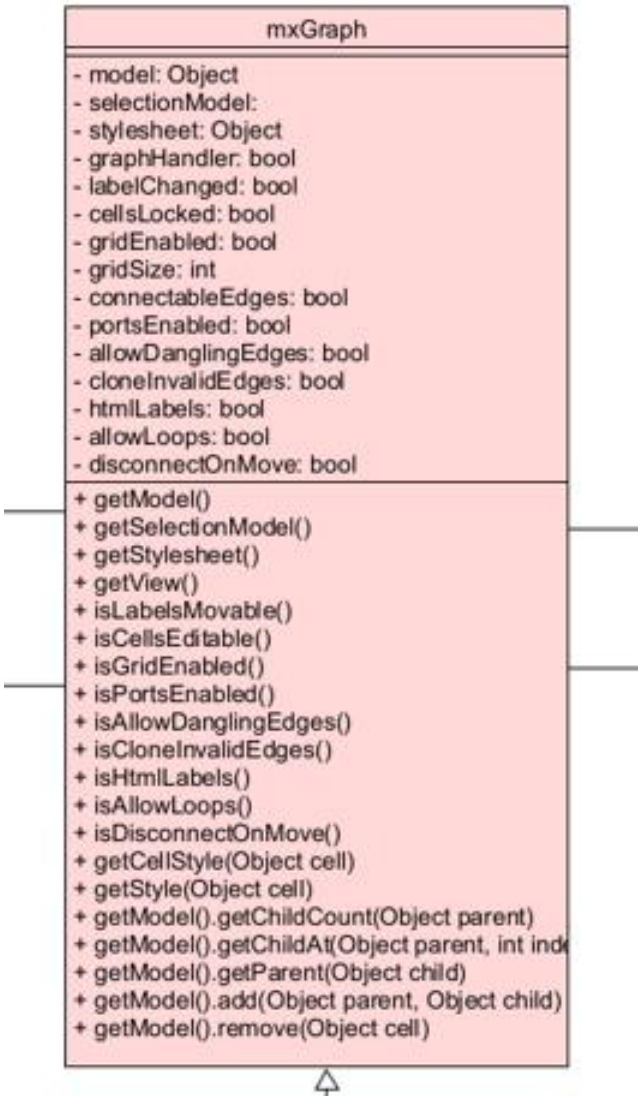
3.1 Diagrama de clases

En el diagrama de clases se puede observar de forma general como está implementado el patrón Compuesto en la librería JGraphX



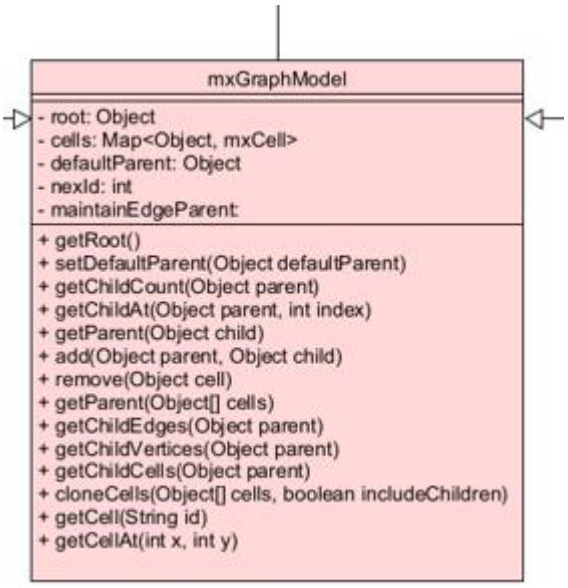
3.1.1 mxGraph

La clase mxGraph en JGraphX es una clase compuesta que implementa el patrón Composite al tener la capacidad de contener nodos individuales y subgrafos, su función es representar un grafo. De esta clase heredan otras clases como mxGraphModel.



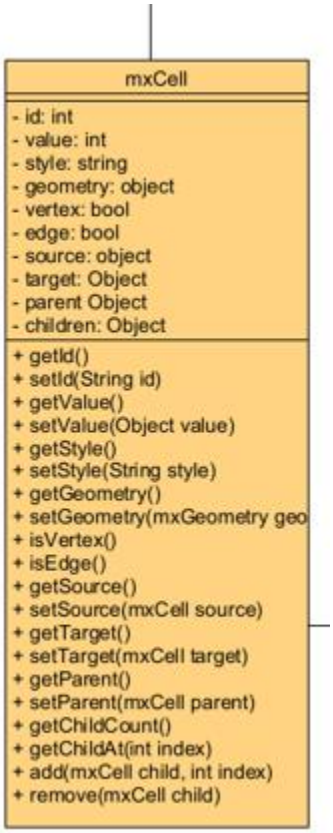
3.2.2 mxGraphModel

La clase mxGraphModel es una clase compuesta que representa un subgrafo y es utilizada por mxGraph para representar la estructura jerárquica del grafo. Además, también realiza operaciones de manejo de inserciones y eliminación de nodos y subgrafos.



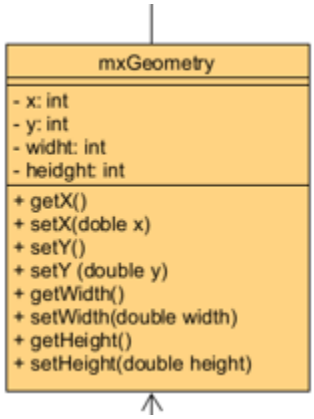
3.2.3 mxCell

La clase mxCell es una clase hoja que representa un nodo o una arista, básicamente representa un elemento individual en un grafo. Además proporciona funcionalidades para almacenar datos, aplicar estilos visuales y gestionar las conexiones con otros objetos de tipo mxCell en el grafo.



3.2.4 mxGeometry

La clase mxGeometry cumple la función de representar la geometría de un elemento (nodo o arista) en un grafo. Esta clase almacena información relacionada con la posición, tamaño y rotación del elemento dentro del grafo.



4.0 Ventajas y Desventajas del Patrón Compuesto en JGraphX

4.1 Ventajas

El uso del patrón composite proporciona varias ventajas para el proyecto y es coherente con los propósitos que se definen para su funcionamiento con respecto al tratamiento uniforme de elementos tanto individuales como compuestos (nodos, vértices, subgrafos, etc..), permitiendo la implementación de operaciones funcionalidades comunes entre elementos. Por otro lado, el patrón también contribuye a la jerarquización de estructuras en el grafo, ya que por medio de la recursividad se componen elementos formando estructuras parte-todo, lo cual facilita el manejo de los datos y permite realizar operaciones recursivas. Todo lo anterior nos lleva a una tercera ventaja, la simplificación y reutilización de código ya que Composite promueva la encapsulación en la estructura del grafo y evita la duplicación de código.

4.2 Desventajas

La principal desventaja que se puede presentar al implementar el patrón Composite en JGraphX es la dificultad en la gestión de cambios, esto gracias a que el uso de estructuras de datos jerárquicas complejas puede dificultar los procesos de agregar, eliminar o modificar elementos en la estructura, por lo que se requiere de análisis rigurosos. Esto puede aumentar el riesgo de introducir errores durante el mantenimiento del código.

Referencias

JGraph. (28 de Octubre de 2020). *JGraphX (JGraph 6) User Manual*. Obtenido de https://jgraph.github.io/mxgraph/docs/manual_javavis.html#2.2.1