Universidad de los Andes
Ingeniería de Sistemas y Computación
Diseño y Programación Orientada a Objetos
Ángel Farfán Arcila-202222183

Link del proyecto: https://github.com/vukan-markovic/Design_Patterns_Paint.git

- En el UML se puede detallar todas las relación que hay entre las clases de este proyecto, se omite los constructores , métodos y propiedades por razones de practicidad, además que no son necesarios para explicar o detallar los patrones de diseño. Adicionalmente, si bien es cierto que el proyecto contiene el patrón observador y se pide no analizarlo también se incluye para mostrar el total del proyecto.

**Nota: Para observar adecuadamente el UML, abrir diagramDeClasesAnalisis.jpg click derecho sobre la imagen y dar Open image in new tab, alli puedes dar zoom y se puede observar bien . Gracias, bonito fin de semestre.

Patrón a evaluar: MVC (Modelo Vista Controlador)

Debe entregar un documento que contenga al menos la siguiente información.

- Información general del Proyecto

    - Este proyecto tiene como finalidad ser una plataforma en la que se puedan diseñar patrones geométricos en los que se puede realizar diferentes figuras las cuales se pueden rellenar de color al gusto del usuario, adicionalmente se pueden guardar las imagenes/patrones creados; funciona como un tipo de plataforma de diseño al estilo Microsoft Paint . La estructura general del proyecto reside en cuatro carpetas principales las cuales son src donde está todo el código , la carpeta docs donde está todo lo que es la información de las imágenes y ejecutables en html, adicionalmente una librería externa en la carpeta lib en la que se encuentra un .jar y el resto son licencias.Uno de los retos que se enfrenta este proyecto de un primer vistazo es la integración de dos patrones de diseño como lo es el MVC y el Observer Pattern.

- Información y estructura del fragmento del proyecto donde aparece el patrón:

    - El patrón MVC se utiliza específicamente en las carpetas que marcan cómo modelo, controller, view.

    - Model:
        - En el modelo se representan los datos de la aplicación.

    - Contiene una lista de ‘shapes’
    - Provee métodos para manipular la lista de figuras.

- Controller:
    - Maneja el input por usuario y actualiza el modelo de acuerdo a esta información.
    - Contiene métodos para adicionar figuras, actualizar formas, mover las figuras, entre otras.
    - Utiliza comandos en cmd para encapsular operación con el fin de hacer uso del control + z en la aplicación o el aclamado undo/redo.
    - Usa el ‘PropertyChangeSupport’ para notificar al View de los cambios.
    - Sirve como enlace entre el modelo y el view.

- View:
    - Se extiende a JPanel y representa la vista gráfica de la aplicación.
    - Es responsable de renderizar las figuras en la pantalla usando el método ‘paint’.
    - Hace referencia mediante el controller del método hallado en el model llamado ‘DrawingModel’ para obtener las figuras para renderizar.

- Información general sobre el patrón:

    - El patrón MVC se utiliza comúnmente en la ingeniería de software como un patrón arquitectural, en el cual se dispone de un orden para estructurar los componentes de un sistema, sus ‘responsabilidades’ y relaciones entre los mismos; lo cual genera bajo acoplamiento. Este se divide en tres capas las cuales son el Modelo , la Vista y el Controlador.

    - Controlador:
        - Los componentes que se encuentran en el controlador sirven para ser intermediario, capturando las interacciones del usuario en la Vista, interpretandolas y generando acciones del sistema en consecuencia. Cuando estas acciones se generan el controlador invoca métodos en el Modelo para consultar o actualizar la información. Por tanto, el Controlador es un coordinador del

sistema MVC, el cual regula interacciones y el flujo de información.

-

- Vista:
    - Esta es la encargada de generar la interfaz de la aplicación, la cual se comunica con el usuario, ya sea mostrando la información o pidiéndola. Ósea contiene elementos de interacción con el usuario, los cuales permiten invocar acciones del sistema que se manejan a través del controller.

- Modelo:
    - Aquí se encuentra la representación de las entidades que nos ayudan a almacenar la información del sistema como las clases de transferencias de datos. También, se encuentran los métodos que representan la lógica del programa el cual está encargado que el programa permanezca íntegro.

- ¿Cómo se está utilizando el patrón dentro del proyecto?

    - El Observer Pattern se está utilizando para gestionar eventos de usuario, como clics del mouse o modificaciones a las figuras. En los cuales se envía una notificación al view para que haga sus respectivas actualización con el controller y el modelo . Esto representa una unión entre dos patrones de diseño en interfaces gráficas y cómo se complementan así mismas.

- ¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

    - Permite que los observadores están desacoplados del modelo, ya que los observadores no necesitan conocer los detalles internos, y permite la extensión y maleabilidad del código.
    - El MVC genera bajo acoplamiento ya que al cambiar la lógica del modelo no se afecta necesariamente al controlador y la vista; y así mismo con los demás componentes del MVC.
    - Se pueden tener diferentes tipos de observadores que respondan eventos/acciones específicas, los cuales los hacen reutilizables.

- El MVC permite la separación de responsabilidades que permite la reutilización de los modelos, además de permitir que el código sea más modular y por tanto más fácil de corregir errores a posteriori.

- ¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?
  - El bajo acoplamiento y la asignación de responsabilidades puede ser compleja de entender. Esto genera una exceso de ‘flexibilidad’ del proyecto lo cual puede llevar a que se hagan abstracciones innecesarias generando así dificultades tanto en el entendimiento como en el rendimiento del proyecto.

- ¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?
  - Eliminando el patrón observador y asignar más carga de responsabilidades al view ;para el desarrollo de la interfaz en Java Swing.