TALLER 5 – DPOO

PATRONES DE DISEÑO

Patrón de diseño: Fachada (Facade)

NOMBRE: ANDRÉS FELIPE PÉREZ MARTÍNEZ

CÓDIGO: 202215659

1. Información General Del Proyecto

   Nombre y URL: El proyecto que se escogió para mostrar el patrón Facade fue "design-pattern-facade" de BrijeshSaxena disponible en la siguiente url: https://github.com/BrijeshSaxena/design-pattern-facade.git.

   ¿Para qué sirve?

   "Design-pattern-facade" es un proyecto educativo cuya principal funcionalidad se centra en permitir la configuración de un sistema de entretenimiento doméstico de una casa para la reproducción de una película. De esta manera, involucra la ejecución de ciertas funcionalidades que se relacionan a la preparación de alimentos (control de uso de electrodomésticos) y al control del sistema de entretenimiento (Espacios de esparcimiento, salas, espacios personalizados, equipos de sonido, equipos de reproducción de video, equipos de ambientación, entre otros equipos electrónicos).

   Algunas de sus más importantes funcionalidades se resumen en:

   - La configuración del sistema de entretenimiento doméstico para reproducir una película.
   - El control de la preparación de alimentos para familiares y amigos.
   - El control del apagado del sistema de entretenimiento en el hogar cuando se completa una película.
   - El control del apagado de los electrodomésticos de cocina después de su uso

   Estructura General del Diseño del Proyecto

   La estructura general del proyecto gira en torno a tres elementos primordiales del diseño que son:

1. **Main**: Es la clase en donde se escribe la aplicación principal que interactúa con el usuario. Además, aquí, se ejecutan los diferentes métodos de la clase HomeFacade, utilizados para la correcta operabilidad de la aplicación y de cada una de sus funcionalidades.
2. **HomeFacade**: Es una clase que, en resumen, se encarga de actuar como el mediador entre la clase principal y los métodos que implementan la lógica del programa. De esta forma, esta clase actúa como una interfaz cuyos métodos abarcan cada una de las funcionalidades que posee el proyecto y que involucran diversas partes de la lógica según el requerimiento solicitado desde la aplicación principal.
3. **Devices**: Es el conjunto de clases que poseen la lógica base del proyecto y que se organizan en paquetes que se convierten en representaciones de los elementos que componen el sistema de entretenimiento de una casa. De esta forma, dentro de estas clases están:

    - Appliance.java
    - Fan.java
    - Light.java
    - SoundBar.java
    - TV.java

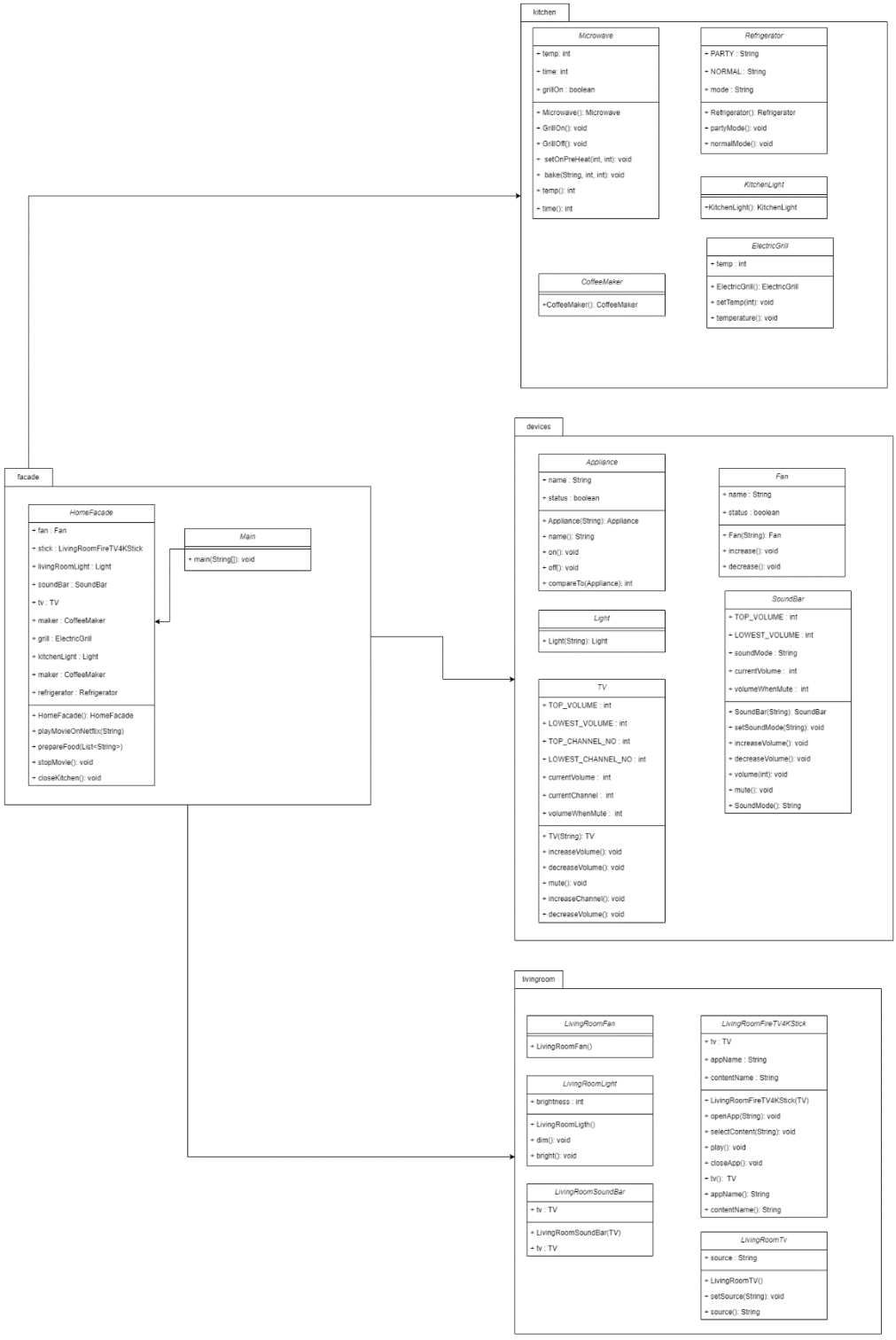
- Kitchen:
    - CoffeeMaker.java
    - ElectricGrill.java
    - KitchenLight.java
    - Microwave.java
    - Refrigerator.java
- Livingroom
    - LivingRoomFan.java
    - LivingRoomFireTV4KStick.java
    - LivingRoomLight.java
    - LivingRoomSoundBar.java
    - LivingRoomTV.java

Finalmente, fijémonos en un pequeño detalle, podemos ver como la estructura general del diseño de este proyecto se basa en un patrón muy parecido al MVC (Modelo-Vista-Controlador) ya que podemos relacionar a Devices con el Model, al HomeFacade con el Controlador y al Main con la vista. Sin embargo, este será un patrón del que no hablaremos en este momento, sino que nos enfocaremos en el patrón Facade, que se roba el papel en la implementación de este proyecto.

Retos de diseño a los que se enfrenta

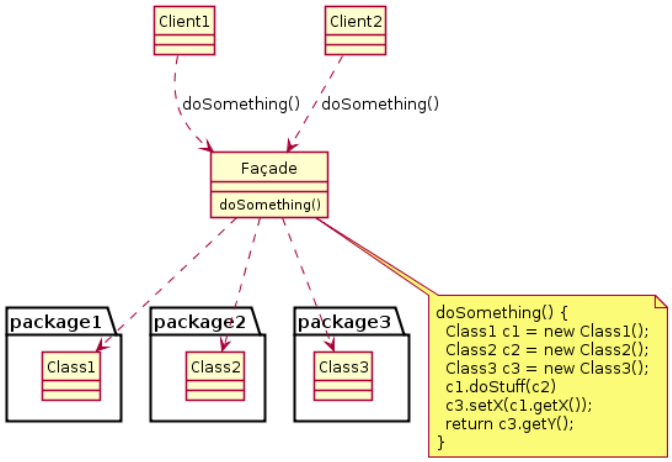
- **Mantener la coherencia con los cambios en el subsistema**: Si el subsistema subyacente (Devices) experimenta cambios o mejoras, puede ser necesario realizar ajustes en la fachada para reflejar esos cambios. Es importante diseñar la fachada de tal manera que sea fácil de mantener y actualizar sin afectar a los clientes que la utilizan.
- A futuro, a medida que se agreguen nuevas funcionalidades y cambios, la interfaz de la fachada debe ser lo suficientemente simple como para facilitar su uso, pero también lo suficientemente completa como para cubrir las necesidades de los usuarios que la utilizarán. Encontrar el equilibrio adecuado puede ser un desafío.
- La alta dependencia que se tiene con la interfaz, es decir, con la fachada. Dado que, si un usuario quiere usar un solo elemento de todo el sistema, tiene que hacerlo mediante el uso de la fachada, que involucra varios elementos del sistema (métodos pertenecientes a varias clases) y lo que puede hacer ver al programa poco eficiente y complicado en términos de acceso a una funcionalidad específica que no involucra más de un componente del programa.

2. El fragmento está estructurado como muestra el siguiente diagrama:



Se puede apreciar la separación en paquetes de las distintas áreas del hogar. Estas, se encuentran unidas a una Fachada en el paquete facade. La cual, simplifica todo para utilizarlo en el main, que finalmente es lo que usara el usuario.

3. Descripción general del patrón de Diseño analizado
   Facade es un patrón de diseño que se basa en el principio de una fachada. Tiene una interfaz que sirve como frente ocultando funcionalidades más complejas. Se utiliza cuando se quiere tener una interfaz simple para reducir la complejidad mediante la división de subsistemas, disminuyendo la dependencia entre ellos, mediante la comunicación a una interfaz principal.



Tomado de: https://en.wikipedia.org/wiki/Facade_pattern

4. En el proyecto “design-pattern-facade” la clase fachada es HomeFacade que ofrece la funcionalidad de las “herramientas que hay en una casa” como, por ejemplo, dispositivos electrónicos, los cuales pueden estar situados en distintos lugares de la casa. HomeFacade hace más fácil el acceso a los dispositivos electrónicos al usuario.

5. Ventajas
   El enfoque promueve un bajo acoplamiento, debido al bajo nivel de interdependencia de los componentes, por ende, los cambios son más fáciles de implementar, ya que pueden ser desarrolladas en cualquier momento sin mayor afectación en otros componentes, además se pueden desarrollar de forma más sencilla las pruebas unitarias. Por último, el sistema es fácil de expandir y mantener, gracias a su acoplamiento suelto.

6. Desventajas

Una de las principales desventajas de haber usado el patrón Facade es la alta dependencia que se tiene con la interfaz, es decir, con la fachada. Dado que, si un usuario quiere usar un solo elemento de todo el sistema, tiene que hacerlo mediante el uso de la fachada. En este tipo de casos es mucho más conveniente usar interfaces o fachadas mucho más especificas en lugar de una fachada global. Por ejemplo, si para el usuario lo mas primordial fuese usar los elementos de la cocina, es contraproducente hacer una fachada global, dado que no se le dará un uso muy grande a los elementos que no hagan parte de la cocina.

7. Otra manera de solucionar el problema que el patrón resuelve es que el main acceda directamente a los métodos de las clases de cada paquete y ejecute la lógica correspondiente para las tareas que cumple el objeto HomeFacade