Taller 5

Juan David Roa Moyano 202211446

Información general del proyecto

El proyecto consiste en un lector de archivos multimedia (mediaFiles). En este se puede reproducir archivos Mp3, y también con herramienta avanzada lee archivos Mp4 y Vlc. El programa imprime por pantalla la canción que esta sonando, y si el formato no es válido el programa lo reconoce y lo imprime como formato no soportado. El reto de este proyecto es que la herramienta Mp3 pueda reproducir el audio de los archivos Mp4 y Vlc.

URL repositorio: <https://github.com/Trebor006/adapter.git>

Autor: Roberto Cabrera

Estructura del proyecto

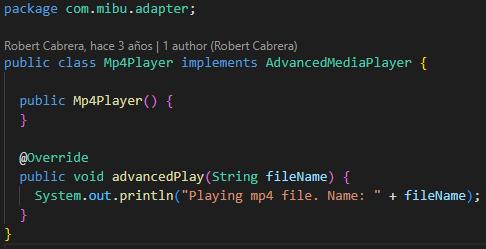
El código contiene 7 clases: un main, 2 interfaces, 3 de las clases de archivos (mp3, mp4, vlc) y uno para reproducir el archivo.

Texto

Descripción generada automáticamente

Clases de archivos:

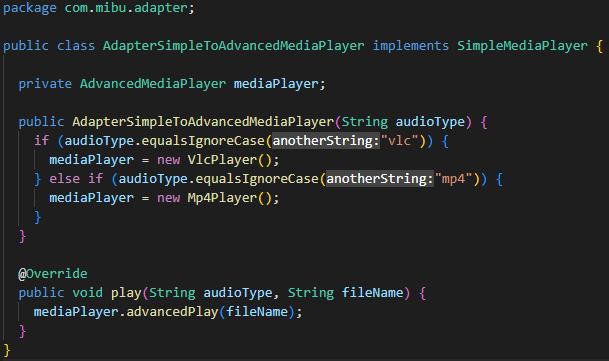




Texto

Descripción generada automáticamente

Como podemos observar el principal archivo multimedia que se lee en el reproductor es Mp3, por ello este contiene un atributo llamado “adapter” de clase AdapterSimpleToAdvancedMediaPlayer. Este atributo es que va a conectar la funcionalidad de leer los otros tipos de archivo.



La clase AdapterSimpleToAdvancedMediaPlayer contiene un atributo llamado “mediaPlayer”, este es el que va a contener el nuevo formato de audio, ya sea Mp4 o Vlc.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Si nos damos cuenta tanto la clase Mp3Player y AdapterSimpleToAdvancedMediaPlayer implementan la interfaz SimpleMediaPlayer que contiene la función de play que es la que reproducirá el audio. Además, también existe la interfaz AdvancedMediaPlayer que puede ser muy similar a la primera interfaz solo que cambia el nombre a advancedPlay.

Para un mayor entendimiento podemos observar el UML del proyecto en la siguiente imagen.

Diagrama

Descripción generada automáticamente

Patrón Adapter

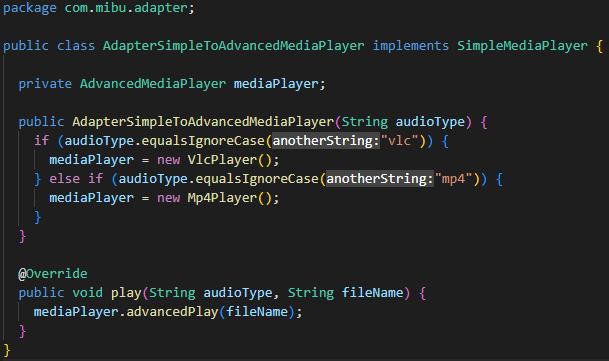
Este patrón consiste en convertir la interfaz de una clase en otra interfaz que los clientes esperan. El adaptador permite que las clases trabajen juntas que de otro modo no podrían debido a interfaces incompatibles.

EL patrón Adapter usualmente se aplica a las siguientes situaciones:

* Cuando quiere utilizar una clase existente y su interfaz no coincide con la que necesita.
* Cuando desea crear una clase reutilizable que coopere con clases no relacionadas o imprevistas, es decir, clases que no necesariamente tienen interfaces compatibles
* Necesita usar varias subclases existentes, pero no es práctico adaptar su interfaz subclasificando a todos. Un adaptador de objetos puede adaptar la interfaz de su clase principal.
* La mayoría de las aplicaciones que usan bibliotecas de terceros usan adaptadores como una capa intermedia entre la aplicación y la biblioteca de terceros para desacoplar la aplicación de la biblioteca. Si se tiene que usar otra biblioteca, solo se requiere un adaptador para la nueva biblioteca sin tener que cambiar el código de la aplicación.

Utilización del patrón

En el proyecto el patrón se puede identificar en la clase AdapterSimpleToAdvancedMediaPlayer. Como podemos ver el propósito de esta clase es leer los archivos multimedia que no eran principalmente utilizados en el reproductor.



Esta clase esta conectando dos interfaces que son la SimpleMediaPlayer y AdvancedMediaPlayer. Como las dos interfaces son dirigidas a clases diferentes de archivos de multimedia, que a la vez se pueden considerar subclases, necesitamos que se adapten al formato de lectura, por ello esta clase tiene como atributo la interfaz AdvancedMediaPlayer.

¿Por qué este patrón?

Como sabemos el problema a solucionar en este código era la lectura de archivos Mp4 y Vlc, desde este punto se necesitaba conectar las nuevas clases correspondientes a estos tipos de archivo sin necesidad de editar el código que ya teníamos. Por ello se opta unir las nuevas clases y su interfaz con la clase AdapterSimpleToAdvancedMediaPlayer, para que se pudiera reutilizar la interfaz que teníamos de base en las siguientes clases a usar. En pocas palabras se necesitaba algo que conectara las interfaces incompatibles y las clases que no se relacionaban.

Ventajas:

* Reutilización de código: El patrón Adapter permite reutilizar el código existente al adaptar una interfaz incompatible en lugar de tener que modificar el código original. Esto puede ahorrar tiempo y esfuerzo en el desarrollo de software.
* Integración sin problemas: El Adapter facilita la integración de componentes o sistemas que utilizan interfaces diferentes. Puede adaptar una interfaz existente para que sea compatible con otra interfaz sin afectar el código existente.
* Flexibilidad: El patrón Adapter proporciona flexibilidad al permitir que diferentes clases trabajen juntas sin requerir cambios en su implementación. Esto permite una mayor modularidad y extensibilidad del código.
* Encapsulación: El Adapter encapsula la lógica de adaptación en una clase separada, lo que ayuda a mantener un código limpio y organizado. Esto también facilita la comprensión y el mantenimiento del código.

Desventajas:

* Complejidad adicional: La introducción de un adaptador puede aumentar la complejidad del código, ya que se agrega una capa adicional de indirección. Esto puede dificultar la comprensión y el mantenimiento del código en algunos casos.
* Sobrecarga de rendimiento: El uso del patrón Adapter puede introducir cierta sobrecarga de rendimiento debido a la necesidad de adaptar las llamadas y convertir los datos entre las interfaces. Sin embargo, en la mayoría de los casos, esta sobrecarga es insignificante y no afecta significativamente el rendimiento general del sistema.
* Posible aumento de acoplamiento: Si no se aplica correctamente, el patrón Adapter puede resultar en un aumento del acoplamiento entre las clases y las interfaces. Esto puede dificultar la modificación o extensión futura de las interfaces o clases involucradas.