Práctica 04: RockBuster

Equipo: SQLazo

Integrantes

- César Becerra Valencia (322064287)
- Víctor Abraham Sánchez Morgado (322606003)
- José Luis Cortés Nava (322115437)

Descripción del Proyecto

Este proyecto es una solución a la práctica 04 de **Modelado y Programación** de la Facultad de Ciencias (UNAM). El objetivo es implementar un sistema para una empresa llamada **RockBuster**, la cual ofrece películas para rentar a sus clientes, además de haber absorbido una cadena de tiendas llamada *Mixdown*, que ofrece discos musicales.

Los patrones de diseño trabajados en esta práctica son el patrón **Adapter** y el patrón **Composite**.

1. Dos maneras de compilar y ejecutar este proyecto

1.1. Con JDK

Prerrequisitos

Este proyecto se puede compilar y ejecutar si se tiene instalado el **JDK (Java Development Kit)** versión 17 o superior.

Cómo compilar y ejecutar

Puedes compilar y ejecutar el programa directamente desde la terminal usando los comandos `javac` y `java`.

Asegúrate de estar en el directorio raíz del proyecto (`Practica03_SQLazo/`) antes de ejecutar los siguientes comandos.

1. Compilación El siguiente comando compilará todos los archivos `.java` que se encuentran en el directorio `src/` y dejará los archivos `.class` compilados en un nuevo directorio llamado `out/`.

```
javac -d out **/*.java
```

- **javac**: Es el compilador de Java.

- **-d out**: Le indica al compilador que coloque los archivos compilados (`.class`) en una carpeta llamada `out`.
- ****/*.java**: Indica que se deben compilar todos los archivos `.java` recursivamente.

2. Ejecución Una vez compilado, puedes ejecutar el programa con el siguiente comando:

```
java -cp out mx.unam.ciencias.myp.rockbuster.catalogo.Main
```

- **java**: Es la Máquina Virtual de Java (JVM) que ejecuta el código.
- **-cp out**: Le indica a la JVM que busque los archivos `.class` en el directorio `out`.
- **mx.unam.ciencias.myp.rockbuster.catalogo.Main**: Es el nombre completamente calificado de la clase que contiene el método `main`.

1.2. Con Docker

Prerrequisitos

Para ejecutar el programa con Docker es necesario tener instalado **Docker Desktop** y mantenerlo abierto durante la ejecución.

Para instalarlo en Ubuntu: `https://docs.docker.com/desktop/setup/install/linux/ubuntu/`

1. Descargar la imagen El comando para construir la imagen estando en el directorio raíz de la práctica es el siguiente:

```
docker build -t rockbuster .
```

- **docker build**: Indica a Docker que debe construir una imagen en base al Dockerfile en la raíz del proyecto.
- **-t rockbuster**: Etiqueta la imagen con el nombre `rockbuster`.
- **.**: Indica a Docker que use los archivos del directorio actual.

2. Ejecutar el contenedor Este comando ejecuta un contenedor basado en la imagen creada:

```
docker run --rm -it rockbuster
```

- **docker run**: Crea y ejecuta un contenedor.
- **rockbuster**: Nombre de la imagen base.
- **–rm**: Elimina el contenedor al finalizar la ejecución.
- **-it**: Hace el contenedor interactivo y con entorno TTY (para usar `Scanner` en la terminal).

2. Anotaciones sobre la implementación de los patrones

Los patrones de diseño utilizados buscan lograr la **extensibilidad** y el **desacoplamiento** del código.

Adapter

Dado que tenemos una interfaz `Product` que define el comportamiento de un producto de la tienda RockBuster, al absorber la compañía Mixdown fue conveniente utilizar este patrón para que los discos pudieran ser tratados como productos de RockBuster sin modificar su estructura interna.

El patrón Adapter consiste en crear una clase que implemente una interfaz específica y contenga una instancia de otra clase existente. Esta nueva clase delega las llamadas a la instancia interna, adaptando su comportamiento a la interfaz requerida.

En este proyecto:

- La clase original: `Album`
- La clase adaptadora: `AlbumAdapter`, que implementa la interfaz `Product` y contiene una instancia privada de `Album`.

Composite

El patrón Composite permite estructurar objetos jerárquicamente, de forma que tanto los objetos simples (hojas) como los compuestos se traten de la misma manera. Los nodos compuestos pueden contener otros objetos del mismo tipo, formando estructuras similares a árboles.

En este proyecto, el patrón se aplica a los objetos que implementan la interfaz `Product`:

- **Movie**: Objeto hoja con propiedades como nombre, creador, género, duración, descripción y costo.
- **Saga**: Nodo compuesto que puede contener tanto películas como más sagas.
- **AlbumAdapter**: Convierte los discos en nodos hoja compatibles con la interfaz `Product`.