Tecnológico de Monterrey Campus Querétaro

Programación orientada a objetos

Proyecto integrador

Maestra:

Silvana De Gyvés Avila

Presentan:

Pablo Javier Arreola Velasco (A01747824)

11 de junio

Índice

Introducción

En los últimos años, se han vuelto populares los servicios de streaming bajo demanda de bajo costo como Netflix, Disney o DC. Algunos de estos servicios se centran en el volumen de videos que se ponen a disposición de los usuarios, mientras que otros tienen el desafío de mostrar solo su propio contenido de marca. El potencial de cada uno de los diferentes tipos de streaming depende del contenido que ofrecen, pero lo que es invariable es la necesidad de estas plataforma de organizar su contenido de la manera más eficiente. Es por lo anterior que he tomado la iniciativa de proveer de una base en forma de programación orientada a objetos para facilitar el diseño de plataformas de streaming.

Diagrama de clases UML

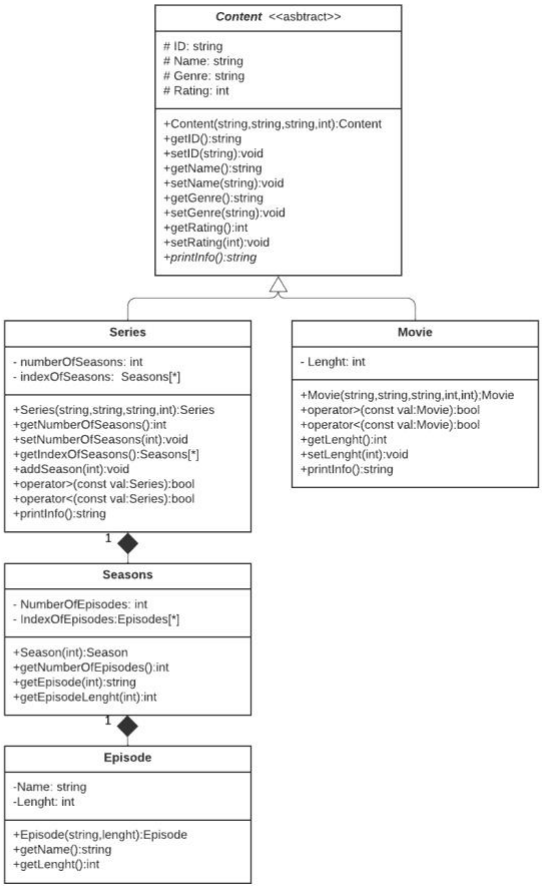

Figura 1: Concepto en forma de diagrama UML a implementar para resolver la problemática

Justificación: Se utilizaron tres clases principales para la herencia: Content (abstracta y padre) que le da forma y funcionalidad a las clases hijas (Series y Movie), estas últimas son el tipo de contenido que se encontraría con facilidad en cualquier página web, para el caso de Series se incluyeron por medio de agregación Season y Episode, Season comprende el índice de objetos tipo episodio que, a su vez, contienen el nombre y la duración de cada uno. Este diseño responde y da solución a la problemática

Ejecución del programa:

Figura 2: Ejemplo de ejecución cuando el usuario ingresa 2 temporadas e imprime la información de la serie

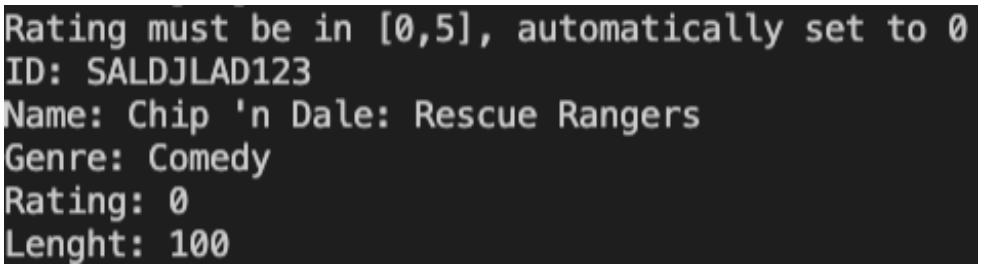

Figura 3: Ejemplo de ejecución cuando el usuario ingresa datos erroneos para una película e imprime la información

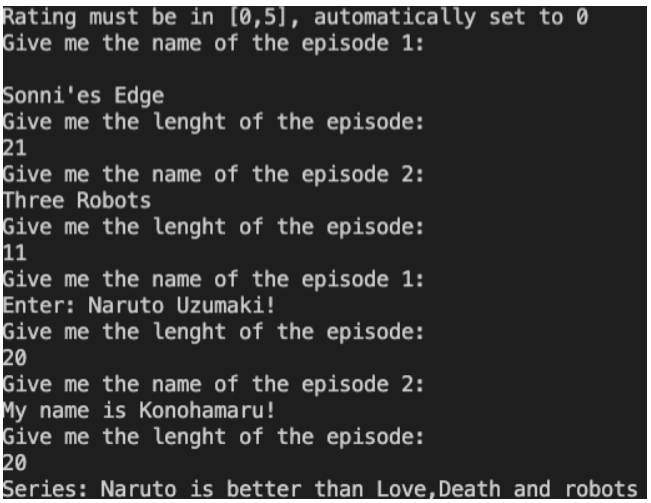

Figura 4: Ejemplo de ejecución cuando el usario define dos series (una con un rango mayor al permitido), utiliza el operador relacional mayor que y descubre que una serie tiene mejor rating que otra

Justificación de decisiones tomadas en el proyecto

A) Las clases Content, Series y Movies incorporadas para este proyecto son las clases indicadas para la resolución de la problemática ya que cada una cuenta con atributos y métodos necesarios para facilitar la implementación de contenido en una plataforma de streaming hipotética. La clase Content proporciona una estructura general para las clases Series y Movies, las clases series por su parte son el tipo de contenido que uno esperaría encontrar en un servicio de streaming, por último las clases Seasons y Episode sirven para agrupar el contenido de series y proveer de una mejor estructura de datos.

B) La herencia implementada tuvo que ver con las clases content, series y películas. La clase base del proyecto es content y las clases derivadas series y películas. La herencia fue correctamente implementada ya que, como se dijo anteriormente, nuestra clase base Content que le otorga las propiedades y métodos a las demás clases, y nuestras clases derivadas, que son especializaciones de contenido, reescriben algunos métodos de content para satisfacer las necesidades que deben de satisfacer. Por ejemplo, si yo quisiera saber la información de una película estaría buscando información distinta a la que estaría buscando sobre una serie, pero aún así habría información de interés que se compartiría entre una y otra.

C) Pasandonos al lado de los modificadores de acceso, estos están correctamente implementados debido a que el acceso a las atributos y métodos fueron pensados teniendo en cuenta funcionalidad y seguridad, dada la distribucón de modificadores de acceso solo las clases hijos pueden acceder a los atributos de la clase padre sin problema pero esto no sería posible desde cualquier otra clase, series y películas protegen de modificaciones sus atributos puesto que solo se pueden manipular por medio de métodos de la misma clase y las temporadas y episodios siguen el mismo comportamiento pero eso no significa que su modificación se complique. Todas las clases tienen sus métodos en público porque no tendría ningún sentido implementar un método al que no vamos a poder acceder.

D) Para el caso de la sobrecarga se realizo una sobrecarga de los operadores relacionales mayor que y menor que, muchas veces es de interés para la clasificación de contenido determinar que serie es mejor puntuada que otra ya que basado en esto se pueden dar recomendaciones al usuario, por eso es importante ayudar a quien sea que diseñe la plataforma a disernir con facilidad.

E) Como se explicó anteriormente algunos métodos de las clases se adaptaron a las necesidades que se tendrían para cada caso, de esta forma el polimorfismo de este proyecto aumentó la funcionalidad invididual de las clases.

F) En el caso de mi proyecto la clase content fue mi clase abstracta y esta se implementó de manera correcta ya que esta fue la que le dio forma a todo el contenido que se encontraría en la plataforma.

G) Por último se introdujo una excepción que protegia a rating de no tener un rango fuera de lo permitido (utilizando throw, catch y try), esta es una forma elegante de proteger los atributos de un uso indebido o que podría causar errores después.

Conclusión

En el desarrollo de este proyecto se utilizaron los conceptos aprendidos en el curso, desde herencia y polimorfismo hasta manejo de excepciones, también reforzamos nuestros conocimientos de buenas prácticas, de diagramas de clases y de debugeo. Aunque se cometieron errores como la implementación de métodos de clase que pedían interacción con el usario también se realizaron muchos avances e investigaciones que llevaron a la implementación de conceptos no tratados en clase. Sin duda creo que el desarrollo de este proyecto me hizo mejorar bastante, ahora cuando veo mis códigos anteriores me doy cuenta de los muchas malas prácticas que tenía, también soy conciente de errores procedimentales que tenía (como el *cowboy coding*), sin duda es un proyecto que me ayudará para el avance de mi carrera.

Referencias

Codingame. (s. f.). Miembros de clase en C++ Variables y Métodos - Clases y Objetos en C++ (Práctica 1). Recuperado 11 de junio de 2022, de https://www.codingame.com/playgrounds/50557/clases-y-objetos-en-c-practica-1/miembros-de-clase-en-c-variables-y-metodos

CG. (s. f.-a). Clases Abstractas e Interfaces - Herencia en C++ (Práctica 3). CodinGame. Recuperado 11 de junio de 2022, de https://www.codingame.com/playgrounds/50747/herencia-en-c-practica-3/clases-abstractas-e-interfaces#:%7E:text=Una%20clase%20abstracta%20es%20aquella,conoce%20como%20clases%20%22concretas%22

CG. (s. f.). Concepto de Herencia - Herencia en C++ (Práctica 3). CodinGame. Recuperado 11 de junio de 2022, de https://www.codingame.com/playgrounds/50747/herencia-en-c-practica-3/concepto-de-herencia

Guzman, H. C. (s. f.). Polimorfismo de clases | Apuntes lenguaje C++ | Hektor Profe. hektorprofe. Recuperado 11 de junio de 2022, de https://docs.hektorprofe.net/cpp/11-clases/06-polimorfismo-clases/