TC1030 (Grupo 832)

Curso Verano 2021

Proyecto Integrador

Ignacio Hernández de la Fuente

A00829933

20/07/2021

Índice de contenido

Introducción

La situación problema nos plantea como objetivo desarrollar una versión simple de una aplicación de apoyo para servicio de streaming. Dicha aplicación debe incluir el uso de una clase base Video, de la cual derivan dos clases, Película y Serie. En base a lo anterior también se pide implementar el uso de herencia y polimorfismo en el código.

En sí, la aplicación debe ser capaz de desplegar en primera instancia un menú que incluya:

a) Una opción que despliegue los videos (tanto películas como series)
b) Una opción que permita calificar algún video (tanto película como serie)
c) Una opción que permita mostrar las series con una calificación determinada
d) Una opción que permita mostrar las películas con cierta calificación
e) Una opción que termine el proceso (que cierre el programa)

En este sentido, el programa que realicé inicializa tres objetos del tipo película y dos objetos del tipo serie, después despliega el menú principal con las 5 opciones mencionadas anteriormente.

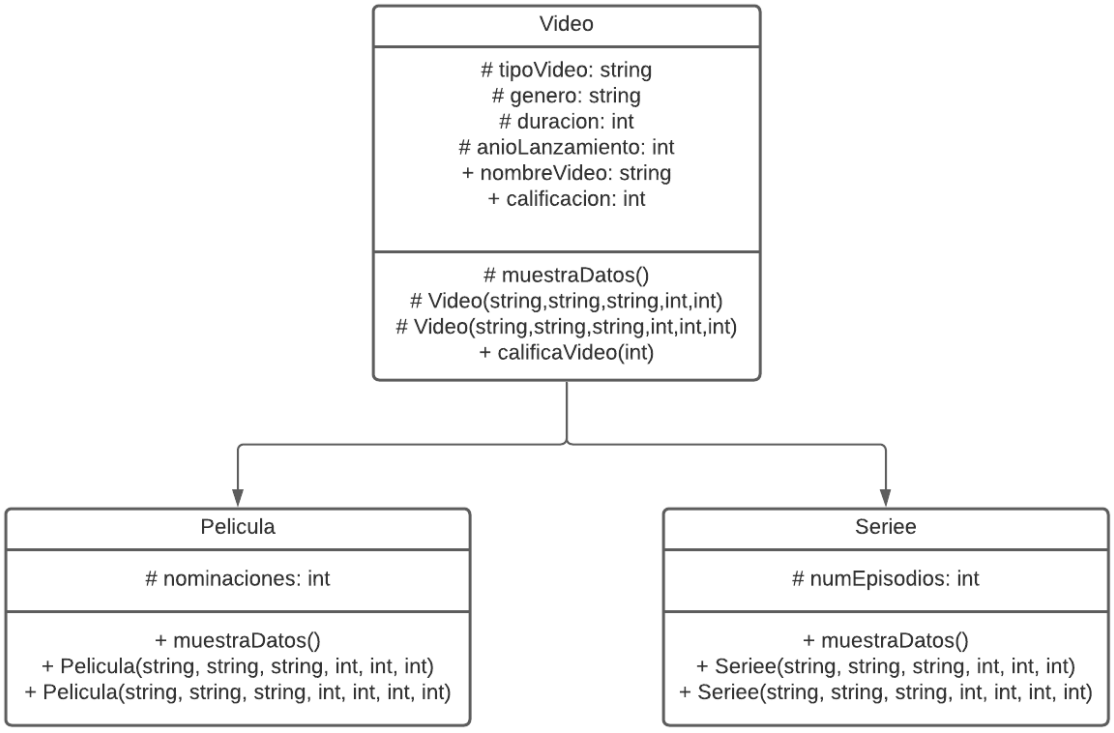
Posteriormente el usuario decide cual opción seleccionar. La primera opción muestra los 5 vídeos y después de mostrarlos el usuario puede seleccionar alguno para ver sus detalles. Si se hace esto el programa despliega los datos del video seleccionado y dará opción para regresar al menú principal.

La segunda opción despliega los 5 vídeos y permite al usuario seleccionar uno para darle una calificación deseada (solo se admiten números enteros entre el 1 y el 5). Automáticamente después de darle la calificación a algún video se regresa al menú principal y se puede corroborar el cambio volviendo a utilizar la primera opción.

La tercer y cuarta opción permiten buscar las series y películas (respectivamente) que tengan la calificación que se introduzca, y la última opción termina el programa.

Con todo lo anterior, el programa realizado cumple con las funciones solicitadas en la problemática propuesta.

Diagrama de clases UML



El diseño del diagrama es bastante simple, se incluyeron las tres clases usadas en el programa; la clase principal Video y sus dos clases derivadas, Película y Seriee. Se da a entender que ambas son clases que derivan de Video por las flechas descendentes. Se utilizó “Seriee” en lugar de “Serie” porque usar la palabra “Serie” provocaba varios errores, posiblemente el sistema lo detectaba como una palabra clave de C++.

En el diagrama se muestran los atributos y los métodos de cada clase, siendo los atributos los que se encuentran en el cuadro debajo del nombre de la clase y los métodos debajo de los atributos. Se especifica el tipo de datos de cada atributo, así como el tipo de dato que debe entrar en cada método (si es un método void no se coloca ningún tipo de dato).

A su vez, se especifica el modificador de acceso de cada método y atributo, siendo los que tienen un “+” los Public y con el “#” se representan los Protected.

Ejemplo de ejecución

Menú principal:

Se proyecta en el momento que se inicia el programa, el usuario puede elegir entre las 5 opciones escribiendo la letra correspondiente.

```
Eliga que desea hacer:
a) Consultar video
b) Calificar algun video
c) Mostrar series con determinada calificacion
d) Mostrar peliculas con determinada calificacion
e) Salir
```

a) Consultar video:

Si se escribe la letra 'a' en el menú principal se proyectarán los nombres de los 5 títulos disponibles, en este apartado el usuario puede ver los detalles de cualquier título deseado escribiendo el número correspondiente.

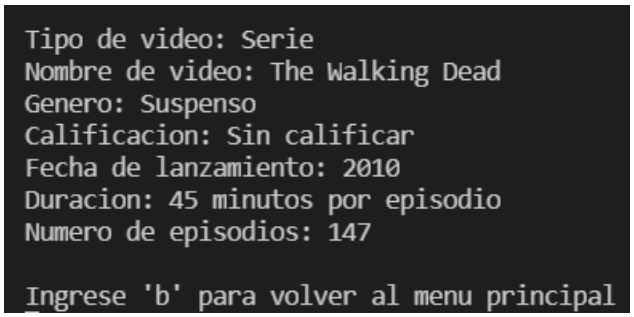
```
1. Requiem For A Dream
2. Shrek
3. Black Widow
4. The Walking Dead
5. Lost
```

Por ejemplo, en la siguiente imagen se muestra lo que el usuario vería si escribiera "2" en el menú anterior. Nótese como la película aún no tiene una calificación (esto se añadirá después).

```
Tipo de video: Pelicula
Nombre de video: Shrek
Genero: Comedia
Calificacion: Sin calificar
Fecha de lanzamiento: 2001
Duracion: 95 minutos
Nominaciones: 2

Ingrese 'b' para volver al menu principal
```

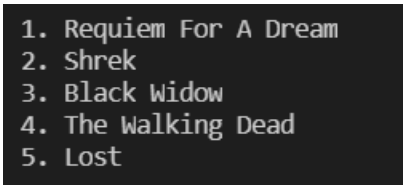
Por otro lado, si el usuario en lugar de “2” escribiera “4”, es decir, que seleccione una serie en lugar de una película, el menú resultante sería el siguiente.



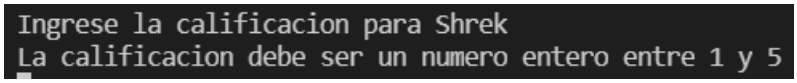
Como se puede apreciar, el tipo de video pasa de ser película a serie, la duración ahora se cuenta con minutos por episodio, desaparecen las nominaciones y aparece el número de episodios.

b) Calificar algún video

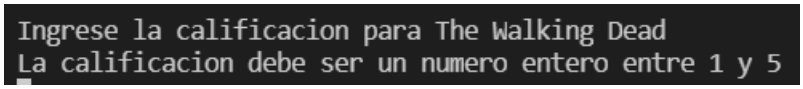
En cualquier caso, el usuario tiene la opción de regresar al menú principal, una vez aquí si el usuario escribe “b” para seleccionar la opción “Calificar algún video”, el menú que aparecerá será el siguiente.



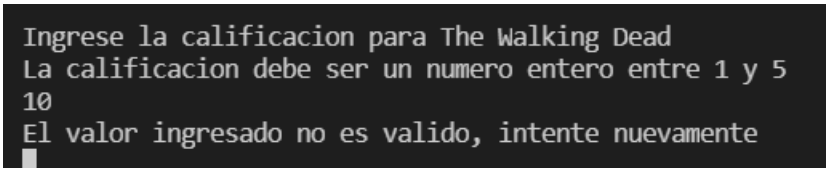
Es idéntico al mostrado si se selecciona la opción “Consultar video”, pero al escribir por ejemplo nuevamente “2”, aparecerá el siguiente mensaje.



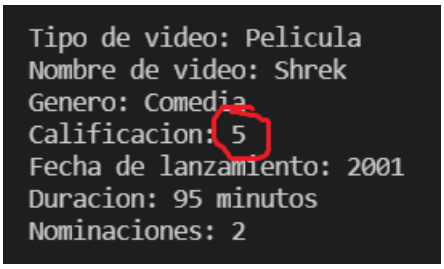
Se le pedirá al usuario la calificación para la película seleccionada, y esta tendrá que ser un número entero entre 1 y 5. El mensaje es igual tanto para series como para películas.



Si se ingresa un numero menor a 1 o mayor a 5, aparecerá el siguiente mensaje. (Se intentó colocar “10” como calificación).



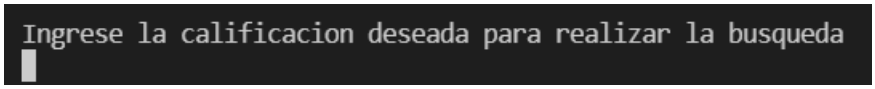
Una vez colocada una calificación válida se regresa automáticamente al menú principal. Desde aquí, seleccionando nuevamente la primera opción, es posible ver el cambio de calificación realizado.



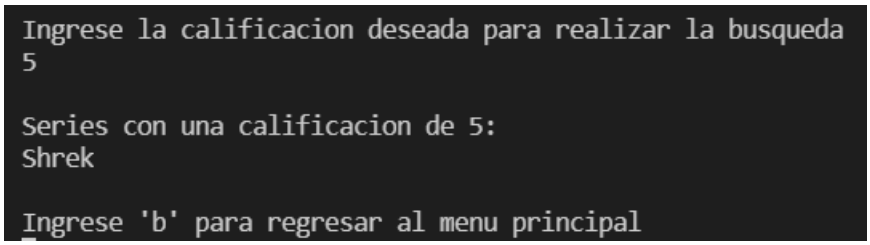
c) Mostrar series con determinada calificación y d) Mostrar películas con determinada calificación

La opción “c” y “d” permiten buscar series y películas (respectivamente) que tengan la calificación deseada.

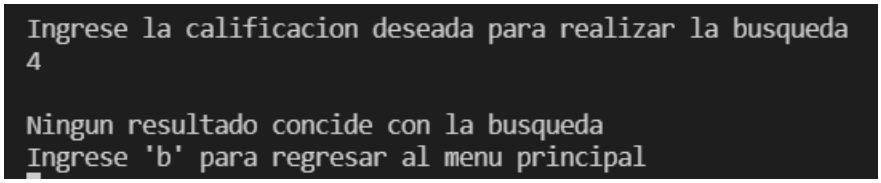
En ambas opciones, el mensaje desplegado será el siguiente.



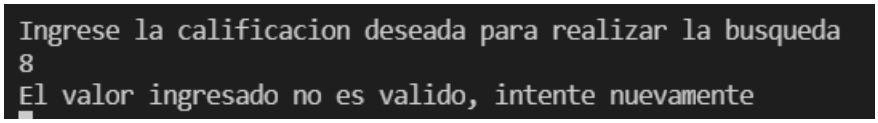
Si se encuentran resultados, se mostrará el siguiente mensaje.



Si no se encuentran resultado, se mostrará el siguiente mensaje.



Por otro lado, si se busca un número invalido (fuera del rango entre 1 y 5) se mostrará el siguiente mensaje.



e) Salir

Por último, si en el menú principal se escribe “e”, es decir, se selecciona la opción Salir, el programa terminará después de desplegar el siguiente mensaje.

Argumentación de las partes del proyecto

En este apartado se explicará de manera clara y concisa como el programa realizado cumple con los siguientes puntos.

a) Se identifican de manera correcta las clases a utilizar.
b) Se emplea de manera correcta el concepto de Herencia.
c) Se emplea de manera correcta los modificadores de acceso.
d) Se emplea de manera correcta el concepto de Polimorfismo.
e) Se explica cómo se podría sobrecargar al menos un operador en el conjunto de clases propuestas.

a) Se identifican de manera correcta las clases a utilizar.

Se solicitó un programa que incluyera las clases Video, Película y Serie. Esto se llevó a cabo de manera exitosa, siendo Película y Serie clases derivada de la clase Video, heredando argumentos y métodos de la clase base, pero también añadiendo nuevos argumentos y métodos exclusivos de cada clase derivada (por ejemplo “número de capítulos” en Serie y “nominaciones” en Película). Se realizó de esta forma para facilitar la realización del código, de hacerse de otra forma (es decir, sin clases) sería mucho más tedioso.

b) Se emplea de manera correcta el concepto de Herencia.

El programa emplea de manera adecuada el concepto de Herencia debido a que las clases derivadas Película y Serie *heredaron* argumentos y métodos de la clase base Video, esto con el fin de ahorrar líneas de código y de facilitar la realización de este, además de evitar potenciales errores. Ejemplos de argumentos heredados en este programa pueden ser “genero”, “tipoVideo”, “nombreVideo”, “calificación”, etc. Así, por ejemplo, tanto los objetos de clase Pelicula como los objetos de clase Serie tienen su propia “calificación” que es independiente de las calificaciones de los demás objetos o clases. Se hizo así para evitar crear múltiples métodos que cumplirían la misma función, complicando más el código.

c) Se emplea de manera correcta los modificadores de acceso.

Dentro de cada clase se hizo uso de únicamente dos tipos de modificadores de acceso, Public y Protected. En la parte Public se colocaron todos los atributos y métodos que se fuesen a usar fuera de la clase, por ejemplo, en el main se mandó a llamar al atributo “calificación” y al atributo “nombreVideo” muchas veces, por eso ambos se encuentran como Public, para facilitar su acceso. Todo aquello que solo se mandase llamar dentro de alguna clase, como por ejemplo los argumentos “genero”, “duración”, “anioLanzamiento”, etc., se incluyeron como Protected. Se utilizó Protected y no Private puesto que Protected permite acceder a los datos desde clases derivadas, y sin esto las clases no podrían acceder a estos datos, lo que complicaría la herencia.

d) Se emplea de manera correcta el concepto de Polimorfismo.

El programa realizado emplea el concepto de Polimorfismo de modo que el método de la clase Video *"muestraDatos()"* incluye un "virtual" antes, lo que permite utilizar dicho método en todas las clases que deriven de Video, modificando a conveniencia lo que queramos que haga en cada clase. En este caso, la clase Serie tiene el método *muestraDatos(),* el cual muestra exactamente lo mismo que el *muestraDatos()* de video, pero se le añade además el número de episodios (característica única de la clase Serie) mientras que la clase Película al usar *muestraDatos()* muestra lo mismo que el *muestraDatos()* de video, pero añade las nominaciones (característica única de la clase Película). Cabe mencionar además que los constructores de las clases derivadas se apoyan del constructor de la clase base. Se realizó de esta manera debido que de otra forma hubiera sido necesario crear múltiples métodos que harían prácticamente lo mismo, además de que no se haría uso del Polimorfismo que se solicitó en un comienzo.

e) Se explica cómo se podría sobrecargar al menos un operador en el conjunto de clases propuestas.

Una posible forma de sobrecargar por ejemplo el operador "+" con la clase Pelicula sería que al usar dicho operador se sumen las duraciones en minutos de las Películas, que se promedien las calificaciones, que se sumen las nominaciones, que se concatenen los géneros, las fechas de lanzamiento, y los nombres de las películas.

Ejemplo:

Pelicula Shrek + Pelicula Black Widow = 229 minutos, 4.5/5, 2 nominaciones, Comedia y Acción, 2001 y 2021, "Shrek y Black Widow".

Con las Series sería similar, pero se sacaría un promedio de duración de capítulos, se sumarian el total de capítulos.

Ejemplo:

Serie The Walking Dead + Lost = 268 episodios, 5/5, Suspenso y Misterio, 2004 y 2010, "The Walking Dead y Lost", 45 minutos promedio por capítulo.

Casos de prueba que causen error

- Seleccionar algún inciso escribiendo la letra en mayúscula en vez de en minúscula (se termina el programa)
- Escribir cualquier otra letra, palabra o numero diferente a los mostrados en pantalla (termina el programa)
- Calificar algún video con numero decimal (termina el programa)
- Al usar las opciones “c” o “d” para buscar videos en base a una calificación, si se escribe cualquier letra o palabra en vez de un numero (crea un bucle infinito de mensajes de texto)

Conclusión personal

Realizar este programa me ayudo a entender mejor la forma en la que se hace un ciclo de múltiples opciones (en este caso en forma de menú principal). También me ayudó a cimentar mejor mis conocimientos sobre herencia y polimorfismo, así como la utilidad que ambas brindan. Me permitió comprender en que casos utilizar Public, Private o Protected y por qué. Me motivó a investigar por mi cuenta conceptos y repasar temas.

Si bien no logré hacer funcionar el código a la primera, considero que no fue tan difícil ni tardado lograr terminarlo. Los principales escenarios donde se me presentaron complicaciones fueron los siguientes.

Al momento de tener que crear un ciclo para hacer el menú. No recuerdo haber hecho uno nunca pero tampoco resultó ser algo tan difícil, simplemente hice uso de while y de if y una vez logré hacerlo funcionar pude seguir adelante sin mucho problema.

Los constructores me llegaron a causar muchos errores, y todo fue porque estaba usando mal la sintaxis al momento de querer emplear el constructor de una clase base sobre el constructor de una clase derivada, para corregirlo volví a ver los ejercicios que hicimos en clase y me basé en ellos.

En un principio muchos errores se presentaban por mandar a llamar variables en Private desde fuera de la clase, por lo que opté por volver TODO Public, pero después me di cuenta de que tampoco era necesario, y terminé poniendo en Protected todo lo que pudiera poner en Protected y puse en Public todo lo demás.

Al hacer la comparación de pelicula.calificacion con la calificación ingresada en las opciones C y D (para buscar series/películas con determinada calificación) se me presentó un error, la calificación aparecía con un numero muy largo que en ningún momento había ingresado. Esto también ocurría con tan solo correr el código y ver los detalles de los videos, salían todos con calificaciones de 6 o 7 dígitos sin sentido alguno. Esto lo corregí estableciendo calificación = 0 desde la clase base. De esta forma eliminé el error tanto en Pelicula como en Serie.