Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Santa Fe

Servicio de Streaming

Situacion Problema

Sebastian Juncos A01022629

Programación Orientada a Objetos, Grupo 300

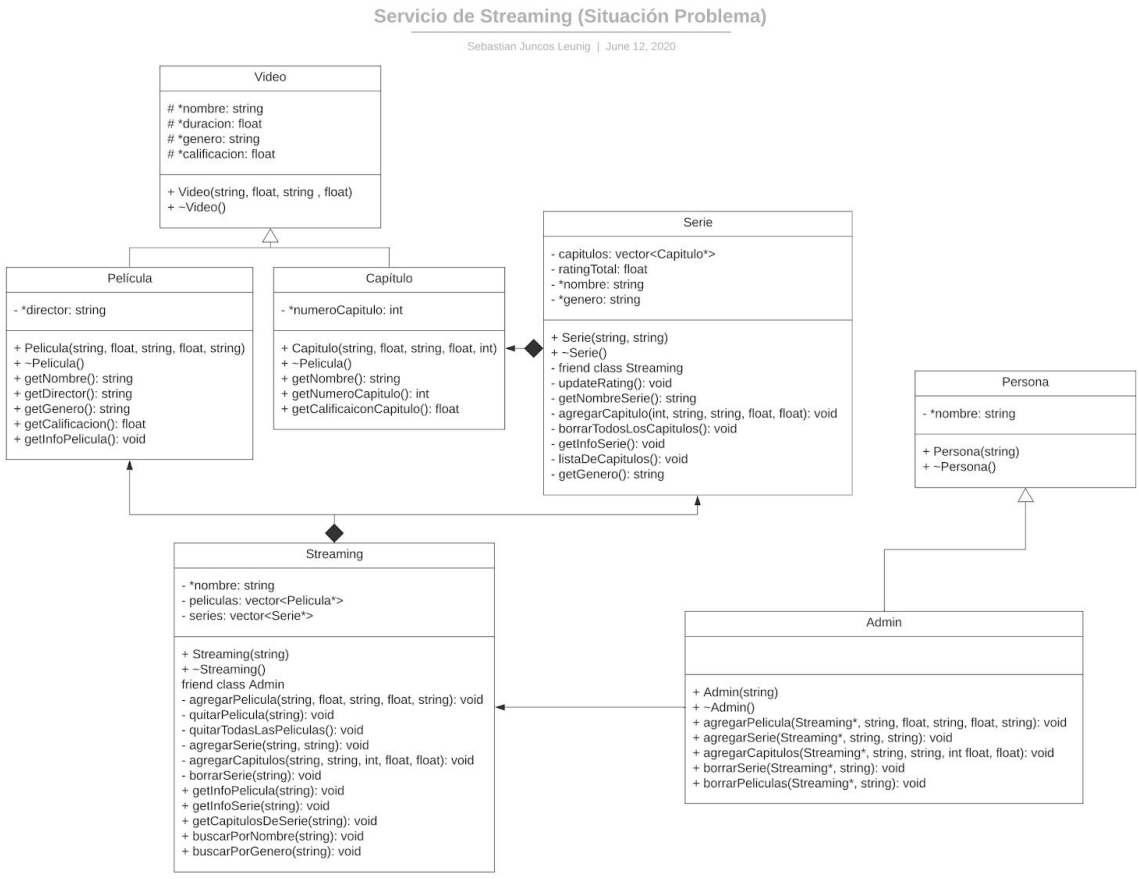
Índice:

SituaciónProblema:

La situación problema pide elaborar una simulación de un servicio de streaming, donde el usuario puede buscar y calificar películas.

Funcionamiento:

El siguiente es mi diagrama de clases para mi solución propuesta:




```cpp
#include <string>
#include <iostream>

using namespace std;
#pragma once

class Video
{
protected:
    string *nombre;
    float *duracion;
    string *genero;
    float *calificacion;

public:
    Video(string, float, string, float);
    ~Video();
```

Primero hice la clase Video, de esta clase heredan la clase película, donde se le agrega el director; y la clase capítulo, donde se le agrega el número del capítulo. Estas clases no tienen nada especial, son simplemente los valores para los videos que contiene el servicio, estos siendo obviamente peliculas y capitulos de series. Ya que todas las variables de estas tres clases son apuntadores y se guardan en heap, todas usan destructores para liberar toda la memoria reservada para cada instancia de estas clases.
Además de esas variables, las clases de capítulo y película tiene getters, para poder usar la información de estas más adelante.

```cpp
#include "Video.h"

#include <string>
using namespace std;
#pragma once

class Pelicula : private Video
{
private:
    string *director;

public:
    Pelicula(string, float, string, float, string);
    ~Pelicula();
    string getNombre();
    string getDirector();
    string getGenero();
    float getCalificacion();
    void getInfoPelicula();
};
```

```cpp
#include "Video.h"


class Capitulo : private Video
{
private:
    int *numeroCapitulo;

public:
    Capitulo(string, float, string, float, int);
    ~Capitulo();
    string getNombre();
    int getNumeroCapitulo();
    float getCalificacionCapitulo();
};
```

```cpp
#include <vector>
#include "Capitulo.h"

class Serie
{
private:
    vector<Capitulo *> capitulos;
    float ratingTotal;
    string *nombre;
    string *genero;
    friend class Streaming;
    void updateRating();
    string getNombreSerie();
    void agregarCapitulo(int, string, string, float, float);
    void borrarTodosLosCapitulos();
    void getInfoSerie();
    void listaDeCapitulos();
    string getGenero();

public:
    Serie(string, string);
    ~Serie();
};
```

Ya con la clase Capítulo, hice un vector de capítulos dentro de una clase llamada Serie, esta clase, como es de esperarse es una serie que automáticamente comparte su género con todos los capítulos dentro de ella. Dentro de esta clase también están las funciones para agregar y quitar capítulos, dado que estas funciones son privadas para que no cualquier usuario pueda

modificarlas, conecté esta clase y sus funciones, con la clase Streaming, haciendo a Streaming una clase amiga de Serie, para poder acceder a sus métodos privados.

```h
#include "Pelicula.h"
#include "Serie.h"
#include <algorithm>
using namespace std;

#pragma once

class Streaming
{
private:
    string *nombre;
    vector<Pelicula *> peliculas;
    vector<Serie *> series;
    friend class Admin;
    // Para las peliculas
    void agregarPelicula(string, float, string, float, string);
    void quitarPelicula(string);
    // Para las series
    void agregarSerie(string, string);
    void agregarCapitulos(string, string, int, float, float);
    void borrarSerie(string);

public:
    Streaming(string);
    ~Streaming();
    // Información
    void getInfoPelicula(string);
    void getInfoSerie(string);
    void getCapitulosDeSerie(string);
    // Busquedas
    void buscarPorNombre(string);
    void buscarPorGenero(string);
};
```

Dentro de la clase Streaming, se encuentra un vector de series y uno de películas, como ya vimos que series tiene también un vector, podemos decir que el vector de series es un vector de vectores. Dentro de esta clase se encuentran todas las operaciones importantes para el manejo del servicio, estos siendo quitar y agregar series, películas y capítulos a las series, la sección de agregar capítulos hace uso de la clase amiga declara dentro de la clase de Serie. Por lo que la clase Streaming administra la creacion y destruccion de capítulos, pero el proceso lo realiza la serie. Siento que esto es más fácil por que cada clase cumple sus funciones, pero las relaciones entre clases, limitan el acceso a estas funciones.

Siguiendo la misma lógica de la relación entre Serie y Streaming, se sigue la misma lógica entre la clase Streaming y la clase Admin, donde la clase Admin es una clase amiga de Streaming, esto sirve para que el administrador sea el único que puede administrar el servicio, tanto los capítulos de serie, como las series y las películas. Esto es más parecido a la realidad, donde existe un servicio como Netflix y los administradores lo manejan.
Además de esto, la clase Admin hereda de la clase Persona, la verdad no tiene nada de especial esta relación simplemente la clase persona es una clase genérica y le da el nombre al administrador.

```h
#include "Persona.h"
#include "Streaming.h"
class Admin : private Persona
{
private:
public:
    Admin(string nombre);
    ~Admin();
    void agregarPelicula(Streaming *, string, float, string, float, string);
    void agregarSerie(Streaming *, string, string);
    void agregarCapitulos(Streaming *, string, string, int, float, float);
    void borrarSerie(Streaming *, string);
    void borrarPelicula(Streaming *, string);
};
```

```h
#include <string>
using namespace std;

#pragma once
class Persona
{
private:
    string *nombre;

public:
    Persona(string nombre);
    ~Persona();
};
```

Ejecución:

El programa tiene 2 tipos de ejecuciones, una es para el Administrador, donde puede agregar y quitar peliculas, agregar y borrar series y meter capítulos en las series existentes.
Por otro lado, un consumidor o “viewer”, puede acceder a la información de películas, series y capítulos, así como buscar por género o nombre.
Las búsquedas deben de ser:

- Por nombre: Para buscar por nombre hay que escribir solo un pedazo del nombre, esto arroja todas las películas y series que contienen esa secuencia de letras dentro del nombre, la búsqueda debe de ser con letras minúsculas.
- Por género: para buscar por género, todas las letras deben de estar en minúsculas, al igual que la búsqueda por nombre, no es necesario introducir el género completo, solo basta con introducir una secuencia, por ejemplo “roman”, para buscar el género de romance.

Por cuestiones prácticas, mi solución propuesta, no cuenta con un menú para que el usuario se mueva libremente dentro de la aplicación, si no que todo se maneja con líneas de código. Pero agregar una interfaz no sería ningún problema para el futuro de este proyecto.
Estas son las líneas que se van a utilizar para el código muestra:

```
Admin *juncos = new Admin("Admin Juncos");
Streaming *netflix = new Streaming("Netflix");
```

Esto nos permite crear al administrador y el servicio de streaming. Después de eso, el administrador (juncos) crea dentro del servicio de streaming (netflix) las series y películas.

```
// Busquedas e informacion:
netflix->buscarPorNombre("w");
netflix->getInfoPelicula("Star Wars");
netflix->buscarPorGenero("drama");
netflix->getInfoSerie("Mirai Nikki");
cout << "\n";
netflix->getCapitulosDeSerie("Mirai Nikki");
cout << "\n";
netflix->buscarPorGenero("roman");
netflix->getInfoPelicula("Tenki no Ko");
cout << "\n";
```

Esta imagen muestra las operaciones que realiza el programa al momento de correrlo:
Buscar por nombre: en este caso vamos a buscar las películas y series que contienen la letra w en ellas.
Después utilizamos
Get Info Pelicula: donde llamamos a la película Star Wars, que también contiene la letra W en el nombre.
Volvemos a realizar un búsqueda y buscamos por el género de drama, en este caso encontramos la serie de Mirai Nikki en la búsqueda, después de encontrar esa serie mostramos la información de la serie y después los capítulos de esta.
Por último, buscamos el género roman, donde la búsqueda autocompleta el género y nos muestra las películas de romance, entre ellas Tenki no Ko. Por lo que después de la búsqueda mostramos la película.
El resultado de estas líneas es el siguiente:

```
Peliculas que contienen 'w':
Kimi No Na Wa
Star Wars
Series que contienen 'w':
No hay series con ese nombre

-------------
¦ Star Wars ¦
-------------
Rating: 4
Director: George Lucas

Peliculas del genero 'drama':
No hay peliculas con ese genero

Series que contienen 'drama':
Control Z
BoJack Horseman

---------------
¦ Mirai Nikki ¦
---------------
Rating: 4.984
26 Capitulos
Lista de Capitulos de Mirai Nikki
1. Sign Up
2. Contract Terms
3. Initial Failure
4. Hand-Written
5. Voice Message
6. Silent Mode
7. Answering Machine
8. New Model
9. Blocking Calls
10. Family Plan
11. Service Terminated
12. No Service Area
13. Number Withheld
14. Memory Erased
15. Double Holder
16. Reapir
17. Family Discount
18. Crossed Lines
19. Clearing Data
20. Transfer Data
21. Security Code
22. Disconnect
23. Unulfilled Contract
24. Searching
25. Reset
26. Format


Peliculas del genero 'roman':
Kimi No Na Wa
Tenki no Ko
Series que contienen 'roman':
Mirai Nikki

---------------
¦ Tenki no Ko ¦
---------------
Rating: 4.8
Director: Makoto Shinkai
```

Posteriormente borramos todo lo guardado en heap y se lo hacemos saber al Administrador:

```
// Borramos todo
juncos->borrarSerie(netflix, "Mirai Nikki");
juncos->borrarSerie(netflix, "One Punch Man");
juncos->borrarSerie(netflix, "Control Z");
juncos->borrarSerie(netflix, "BoJack Horseman");

juncos->borrarTodasLasPeliculas(netflix);
delete netflix;
delete juncos;
```

Como podemos ver, las series tienen que ser borradas una por una, mientras que las películas pueden ser borradas todas con un solo comando. Esa función se podría agregar para las series en algún futuro. Estas líneas nos muestran en la terminal:

```
Se ha borrado la serie 'Mirai Nikki'
Se ha borrado la serie 'One Punch Man'
Se ha borrado la serie 'Control Z'
Se ha borrado la serie 'BoJack Horseman'
Todas las peliculas han sido borradas
```

Conclusión:

Este proyecto me encantó, la verdad fue un gran reto que incluso tuve que empezar desde 0 una segunda vez, pero valió la pena, realmente me ayudó a tener una mejor idea de como utilizar los objetos y la relación entre diferentes clases, además del uso adecuado de los apuntadores. Además de lo aprendido en clase, mi propuesta de solución me ayudó a aprender la función de clases amigas, ya que es un tema que no vimos en clase. Estoy muy contento con lo que aprendí en la clase, con el excelente profesor, con mis resultados y con cómo se llevó a cabo la clase a pesar de la situación en la que nos encontramos.