Programación Avanzada

Laboratorio 0

Consideraciones generales:

● La entrega podrá realizarse hasta el Lunes 22 de abril hasta las 18:00hs.

● La defensa se realizará el mismo Lunes a las 20hs. Los grupos deberán defender el resultado de su implementación. Además, se le brindará una prueba a cada participante del grupo basada en el laboratorio, para esto cada integrante deberá de tener el código en su equipo o de lo contrario no aprobará el laboratorio.

● El código fuente deberá ser entregados mediante el EVA del curso dentro de un archivo con nombre <número de grupo>_lab0.zip (o tar.gz). Dentro del mismo archivo comprimido.

● Las entregas que no cumplan estos requerimientos no serán consideradas. El hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

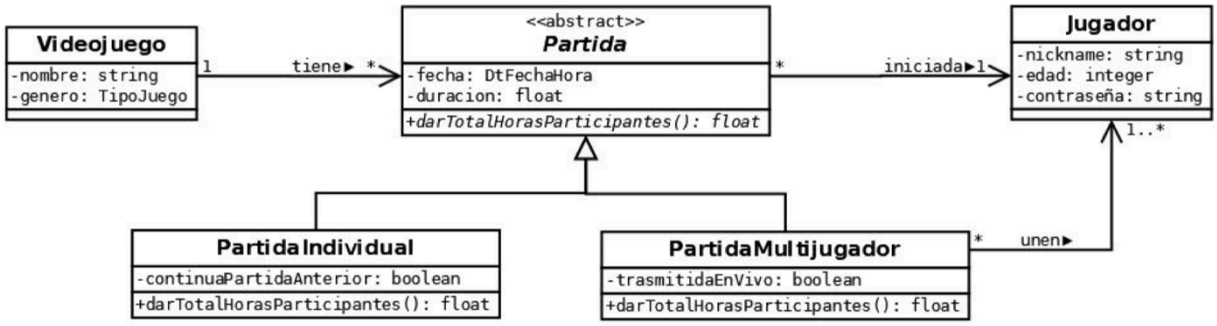
Objetivo

En el laboratorio 0 se espera que el estudiante repase la implementación en el lenguaje C++ de operaciones básicas y otras construcciones del lenguaje, así como el uso/repaso básico del entorno de programación. También se espera que el estudiante consulte el material disponible en el EVA del curso y complemente las consultas recurriendo a Internet con espíritu crítico y corroborando.

Problema

Se desea implementar un prototipo de sistema para aficionados a los videojuegos, el cual debe contar con un catálogo de videojuegos de diversos géneros, disponibles exclusivamente para los jugadores **registrados en el sistema.**
De cada videojuego se registra su nombre, el cual lo identifica, y su género. Los jugadores se registran ingresando un nickname (que lo identifica), edad y contraseña, y pueden iniciar tanto partidas individuales como multijugador.
En estas últimas, siempre participa más de un jugador y pueden ser transmitidas en vivo.
El sistema almacena las partidas jugadas registrando la fecha en la que se realizan y su duración (en horas).

Para cada partida individual interesa saber además si la misma fue una partida nueva o una continuación de una anterior.
Por último, es de interés contar con estadísticas que permitan conocer cuáles son los videojuegos preferidos por los usuarios. Para esto, el sistema permite conocer la cantidad total de horas dedicadas por los jugadores a determinado videojuego.
En base a la descripción anterior se ha diseñado la estructura de clases que muestra la siguiente figura.



Ejercicio 1

Se pide implementar en C++:

1. Todas las clases (incluyendo sus atributos, pseudo-atributos, constructores y destructores, y los getters y setters necesarios para las operaciones que se indican más adelante), enumerados y datatypes que aparecen en el diagrama.
Para las fechas en caso de recibir dd > 31 o dd < 1 o mm > 12 o mm < 1 o aaaa < 1900, se debe lanzar la excepción. No se deben hacer más que estos controles en la fecha (ej. la fecha 31/2/2000 es válida para esta realidad).

2. Las siguientes operaciones:

a) void agregarJugador(string nickname, int edad, string contraseña)

Registra un nuevo jugador en el sistema. Si ya existe un jugador registrado con el mismo nickname, se lanza una excepción.

b) **void agregarVideojuego(string nombre, TipoJuego genero)**

Registra un nuevo videojuego en el sistema. Si ya existe un videojuego registrado con el mismo nombre, se lanza una excepción.

c) **Jugador* obtenerJugadores(int cantJugadores)**

Devuelve un arreglo con información sobre los jugadores registrados en el sistema. El parámetro cantJugadores es un parámetro de salida donde se devuelve la cantidad de jugadores devueltos por la operación (corresponde a la cantidad de instancias de Jugador retornadas).

d) **VideJuego* obtenerVideojuegos(int cantVideojuegos)**

Devuelve un arreglo con información sobre los videojuegos registrados en el sistema.
El parámetro cantVideojuegos es un parámetro de salida donde se devuelve la cantidad de jugadores devueltos por la operación (corresponde a la cantidad de instancias de Videojuego retornadas). El atributo totalHorasDeJuego corresponde a la suma de horas jugadas por todos los jugadores del videojuego. Esto debe calcularse sumando las horas devueltas por las invocaciones a **darTotalHorasParticipantes**() sobre cada partida del juego. Si la partida es individual, la cantidad de horas devuelta por dicha operación es igual a su duración, mientras que si es multijugador devuelve su duración multiplicada por la cantidad de participantes de la partida.

e) **Partida* obtenerPartidas(string videojuego, int cantPartidas)**

Devuelve un arreglo con información de las partidas del videojuego identificado por videojuego. El parámetro cantPartidas es un parámetro de salida donde se devuelve la cantidad de partidas devueltas por la operación (corresponde a la cantidad de instancias de Partida retornadas). Entre los datos devueltos para ambos tipos de partida se encuentra la duración. Además, entre los específicos de cada partida individual se encuentra si la misma es o no continuación de una partida anterior, mientras que para cada partida multijugador se indica si la misma es transmitida en vivo y la cantidad total de jugadores que se unen a la misma junto con sus nicknames. Si no existe un videojuego registrado con el nombre enviado, se lanza una excepción.

f) **void iniciarPartida(string nickname, string videojuego, Partida* datos)**

Registra una partida individual o multijugador del videojuego identificado por videojuego, iniciada por el jugador identificado por nickname. El parámetro de entrada datos contiene la información completa de la partida. Entre los datos comunes a ambos tipos de partida se encuentra la duración. Además, si datos es una instancia de PartidaIndividual contiene si la partida es o no una continuación de una partida anterior, mientras que si es un instancia de PartidaMultijugador, indica si la partida es transmitida en vivo y la cantidad total de jugadores que se unen a la misma junto con sus nicknames. La partida se da de alta con la fecha del sistema al momento del registro. Si no existe un jugador o videojuego registrado con el nickname y nombre enviados, se lanza una excepción.

Notas:

● Puede implementar operaciones auxiliares en las clases dadas en el diagrama si considera que le facilitan para la resolución de las operaciones pedidas.

● Se puede utilizar el tipo std::string para implementar los atributos de tipo string.

● En este laboratorio se pueden utilizar estructuras de datos de la biblioteca STL, tales como vector, set, map, etc.

● Se sugiere crear una clase auxiliar llamada **Sistema** que implemente las operaciones pedidas y almacene el conjunto de jugadores y videojuegos.