Programación Avanzada

EXAMEN JULIO 2018

- Utilice lápiz y escriba con letra clara y legible.
- Escriba su nombre y número de documento en todas las hojas que entregue.
- Numere las hojas e indique el total de hojas en la primera de ellas.

Problema 1 (34 puntos)

Una empresa que organiza eventos desea un sistema que le permita la gestión de los mismos. Los eventos se identifican con un código, tienen un costo fijo asociado y se desea registrar si son provistos directamente por la empresa o a través de una tercerización. De cada evento se conoce además su fecha de realización, la cantidad de asistentes y el lugar donde se realizará. Existen varios tipos de eventos más específicos: Familiares, Empresariales y Académicos. Para el caso de los Empresariales se registra el número de RUT de la empresa y su razón social; si es Académico, la institución y si es de carácter internacional o no.

Todo evento tiene un responsable de su organización, que trabaja en la empresa, y una persona contratante. Una persona contratante no puede ser a la vez responsable del mismo evento. Las personas son identificas por su cédula de identidad; además se conoce de ellas su nombre, apellido y fecha de nacimiento.

Un evento puede incluir varios servicios. Los mismos son de diferentes tipos: Conferencias, Ambientación del lugar, Servicio de catering, Recreación, Música. Las Conferencias serán dadas por uno o más expositores, personas de las cuales se desea conocer además su currículum vitae. Estos servicios no son ofrecidos para los eventos Familiares. Para el caso del Servicio de catering se desea conocer una descripción del menú y para el caso de Recreación una explicación sobre qué consiste el servicio.

Caso de Uso	Registrar evento
Actor	Administrador
Descripción	Este caso de uso comienza cuando el administrador desea registrar un nuevo evento. Para ello debe ingresar los datos básicos correspondientes al evento, los específicos correspondientes al tipo de evento y los identificadores del contratante y del responsable del mismo. Luego debe ingresar los servicios asociados al evento, junto con la información correspondiente a cada uno.

SE PIDE:

a) **Modelo de Dominio** de la realidad anterior (considerando el Caso de Uso) con restricciones en lenguaje natural.
b) **Diagrama de Secuencia del Sistema** (DSS) del Caso de Uso, incluyendo el uso de Datatypes y de manejo de memoria del Sistema, en caso de ser necesario.

Problema 2 (34 puntos)

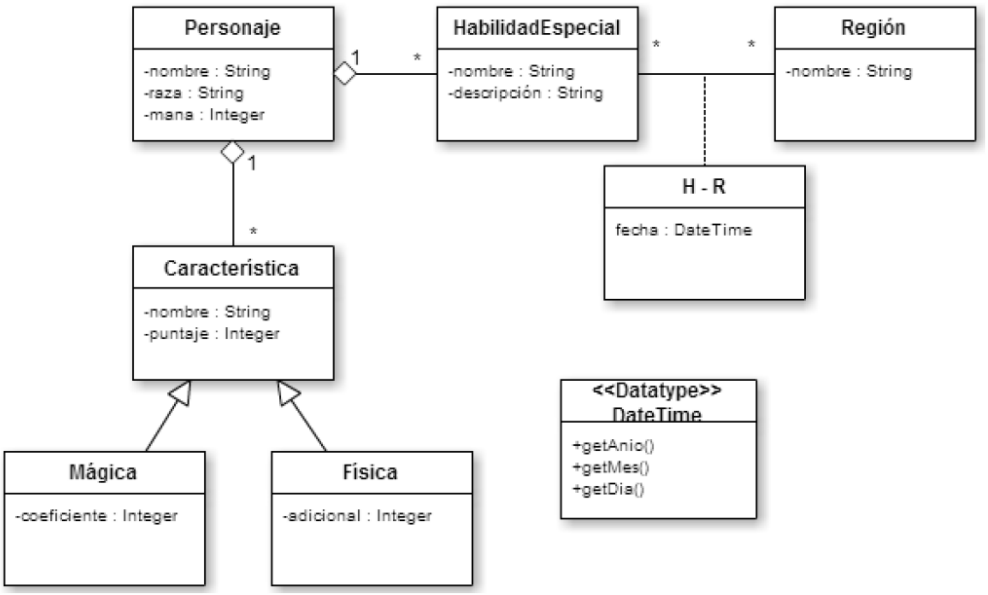
Contexto:

Un juego de rol (traducción típica en español del inglés *role-playing game*, literalmente «juego de interpretación de roles») es un juego en el que, tal como indica su nombre, uno o más jugadores desempeñan un determinado rol, papel o personalidad de un personaje, participante en una historia. En dichos juegos existe lo denominado "Ficha del personaje" que pretende resumir una descripción del personaje que el jugador luego interpretará en la historia, para ser de referencia. A usted y a su equipo de programación, le han designado la tarea de realizar el diseño de un pequeño módulo, de una plataforma destinada al mantenimiento de fichas de los jugadores.

Realidad:

En la plataforma de mantenimiento de fichas, existen los personajes, de los cuales se conoce su nombre, su raza y su reserva de mana (mana). Luego existen las características con su nombre y su puntaje, las cuales pueden ser físicas o mágicas. En caso de ser mágicas interesa saber su coeficiente, mientras que en caso de ser física interesa saber su adicional. Por otro lado, cada personaje puede tener un conjunto de habilidades especiales. Dichas habilidades son adquiridas en distintas regiones del mundo. Es de interés, registrar la fecha en que se adquirió una habilidad en una región determinada.

El equipo de Análisis le ha otorgado el siguiente modelo:



Restricciones:

1) Un *Personaje* se identifica por su *nombre*.
2) Una *Región* se identifica por su *nombre*.

Se consideran los casos de uso "*Aprender habilidad especial*" y "*Obtener información de un personaje*", cada uno de los cuales es modelado con una única operación del Sistema, cuyos contratos se especifican a continuación:

aprenderHabEspecial(nombreP, nombreR, nombreH, descHab**: String,** fechaHR**: DateTime)**	
Descripción	El jugador aprende una nueva habilidad en una región determinada.
Parámetros	- nombreP: nombre del personaje. - nombreR: nombre de la región. - nombreH: nombre de la habilidad aprendida. - fecha: fecha en la que aprende la habilidad en la región. - descHab : descripción de la nueva habilidad.
Precondiciones	- Existe en el Sistema una instancia de Personaje con nombre igual a nombreP. - Existe en el Sistema una instancia de Región con nombre igual a nombreR. - NO existe en el Sistema una instancia de Habilidad con nombre igual a nombreH asociada al personaje de nombre igual a nombreP.
Postcondiciones	- Existe en el Sistema una nueva instancia de HabilidadEspecial con nombre igual a nombreH y con descripción igual a descHab. - Se crea un link entre la instancia de HabilidadEspecial creada previamente, con el personaje de nombre igual a nombreP. - Se crea una instancia de tipo asociativo llamada H-R con fecha igual a fechaHR, entre la instancia de Personaje con nombre igual a nombreP y la instacia de Región cuyo nombre es igual a nombreR.

obtInfoPersonaje(nombreP**: String) : DataPersonaje**	
Descripción	Se retorna información de un personaje.
Parámetros	- nombreP: nombre del personaje.
Precondiciones	- Existe en el Sistema una instancia de Personaje con nombre igual a nombreP.
Postcondiciones	Se retorna un datavalue con la siguiente información del personaje cuyo nombre es nombreP: - Nombre, raza y mana del personaje. - Nombre y descripción de todas sus habilidades especiales. - Los puntos totales del personaje, lo cual se calcula cómo suma total de los puntajes de todas sus características, teniendo en cuenta que en caso de ser una habilidad mágica se contabiliza cómo puntaje * coeficiente, y en caso de ser una habilidad física se contabiliza cómo puntaje + adicional.

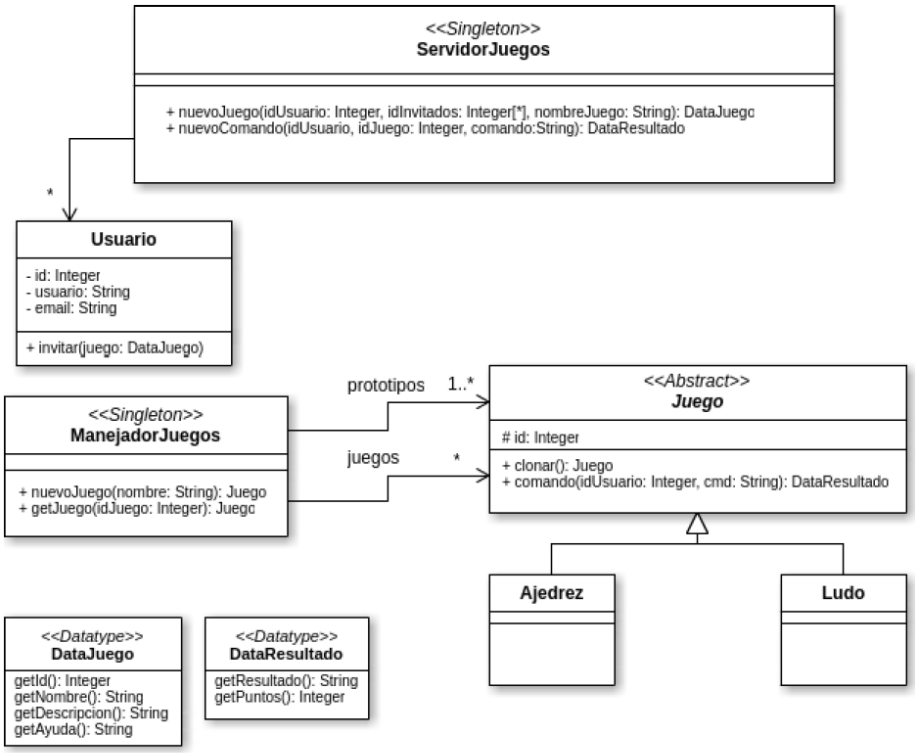
SE PIDE:

a) Realizar el **Diagrama de Comunicación** para cada una de las operaciones previamente declaradas.
b) Realizar el **Diagrama de Clases de Diseño** correspondiente.

Problema 3 (33 puntos)

Se pretende implementar un servidor de juegos en línea, donde un usuario registrado puede invitar a uno o más a jugar una partida de los juegos existentes.

Se diseñó para tal fin un prototipo del sistema, el cual debe implementar. La siguiente figura muestra un diagrama de clases de diseño:



Nota: No se incluyen dependencias

El controlador *ServidorJuegos* cuenta con las siguientes operaciones:

- *nuevoJuego(idUsuario: Integer, idInvitados: Integer[*], nombreJuego: String): DataJuego*
  Genera una nueva instancia del Juego de nombre *nombreJuego*, y retorna sus datos en un datatype *DataJuego*. El usuario con ID *idUsuario* puede invitar a uno o más usuarios registrados en el sistema, cuyos IDs se deben incluir en el arreglo *idInvitados*.

- *nuevoComando(idUsuario, idJuego: Integer, comando: String): DataResultado*
  Mediante esta operación del sistema el usuario identificado por el ID *idUsuario* envía un comando al Juego de ID idJuego.

Mediante la operación *Usuario::invitar* se envía una invitación a un determinado Juego a un Usuario. El singleton *ManejadorJuegos* tendrá una colección de instancias de juegos que utilizará como prototipos para crear otros nuevos. Cuando se invoque su operación nuevoJuego, deberá devolver una instancia nueva del Juego del nombre dado por parámetro, y a su vez agregarla a la colección de juegos en curso. Cada Juego será clonado invocando la operación clonar del prototipo pertinente.

Considerar:

- Puede suponer la existencia de la interface *ICollectible* e implementaciones de *ICollection* (clase *List*), *Ikey* (clase *OrderedKey*), *IDictionary* (clase *OrderedDictionary*) e *IIterator*, según sea necesario.
- Es posible utilizar las clases *set*<*T*>, *vector*<*T*> o *map*<*K,T*> de la STL.
- Las implementaciones deben incluir constructores y destructores.
- Implementar los *getters* solamente para *datatypes*.
- No implementar *setters* y *getters*, salvo la operación *getJuego*.
- No incluir directivas al precompilador.

SE PIDE:

a) Implementar .h y .cpp del datatype *DataJuego*.
b) Implementar .h y .cpp de la clase *ServidorJuegos*.
c) Implementar .h y .cpp de la clase *Usuario*.
d) Implementar .h y .cpp de la clase *ManejadorJuegos*.
e) Implementar .h y .cpp de las clases *Juego* y *Ajedrez*.
f) Las operaciones *invitar* y *comando* deben tener método vacío.