Programación 4

EXAMEN DICIEMBRE 2018

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas.
- 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.
- Recuerde entregar su número de examen junto al examen.
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial.

Problema 1 (35 puntos)

Con la proximidad del verano y aprovechando el buen clima que caracteriza al mes de diciembre, una importante cadena de gimnasios en Uruguay se acercó a su equipo con la siguiente idea: construir una aplicación para conectar personas que quieren hacer ejercicio en su casa con profesores de educación física. Tras algunas reuniones de análisis, se confeccionó la siguiente descripción de la realidad.

De cada usuario interesa conocer su nick que lo identifica, correo y fecha de nacimiento. A su vez, los usuarios pueden ser deportistas, que son quienes buscan ejercitarse, o profesores, que son quienes proponen rutinas de ejercicios para los primeros.

Para cada deportista se conoce adicionalmente su peso actual y un teléfono de contacto en caso de emergencia. En el caso de los profesores, adicionalmente se conoce si se encuentra certificado por algún instituto de educación física o no.

Los profesores plantean rutinas de ejercicios. De cada rutina interesa conocer su nombre, descripción, la cantidad de días que dura la misma y el nivel de exigencia, el cual puede ser: básico, intermedio o avanzado. A su vez, la rutina está compuesta por una sucesión de días (día 1, día 2, … día N). Para cada día de la rutina, el profesor plantea una serie ordenada de ejercicios. De cada ejercicio interesa conocer su nombre, una descripción de ayuda (pasos y consejos para realizar el ejercicio), cantidad de repeticiones, kilo calorías a consumir y el grupo muscular que se trabaja en dicho ejercicio. Por tratarse de la primera versión de la aplicación, se manejarán solamente los siguientes grupos musculares: piernas, brazos, espalda, pecho y abdominales.

Por otro lado, para cada día de la rutina interesa conocer la duración aproximada en minutos de la serie de ejercicios y la cantidad total de kilo calorías a consumir, la cual se calcula como la suma de las kilo calorías a consumir en cada ejercicio previsto para ese día.

Los deportistas se ponen a si mismos a prueba, “desafiando” una rutina de ejercicios. Cuando un deportista comienza una nueva rutina de ejercicios, se quiere guardar la fecha en que comienza el desafío y su peso. Por otro lado, un deportista puede desafiar más de una rutina de ejercicios al mismo tiempo, pero no puede desafiar dos veces la misma rutina.

Cada vez que un deportista completa la serie de ejercicios diarios de una rutina, se guarda la fecha en que lo hizo. Además y de forma opcional, el deportista puede ingresar su peso actual. De esta forma el deportista puede conocer la evolución de su peso día a día.

Finalmente, cuando el deportista completa todos los días de una rutina, se considera que terminó el desafío, registrándose la fecha en que termina y su peso final.

Se relevó además el siguiente caso de uso:

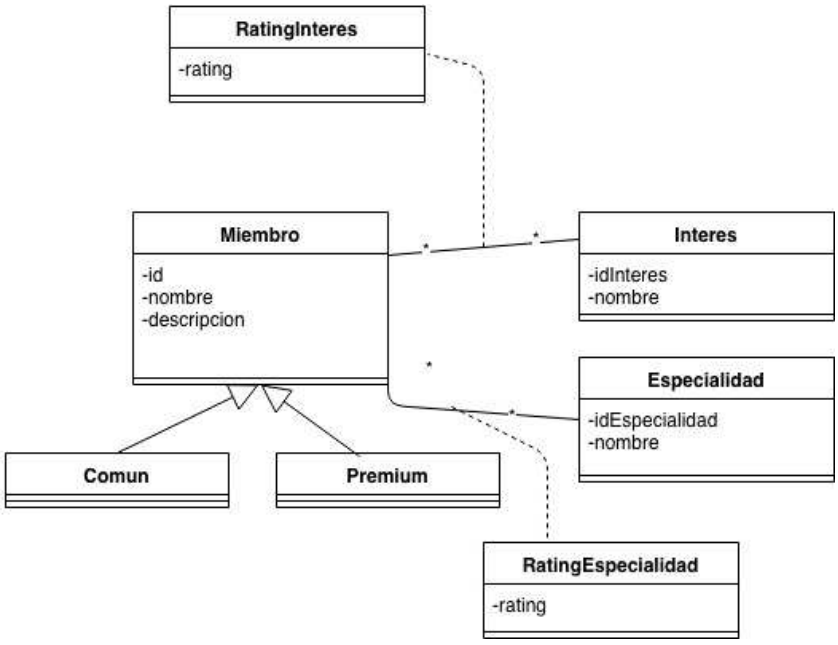
Caso de Uso	Crear Rutina
Actor	Profesor
Descripción	El caso de uso comienza cuando un profesor que se encuentra con una sesión iniciada en el sistema, indica que desea plantear una nueva rutina de ejercicios. Para ello el sistema le pide que ingrese la información básica de la rutina, indicando nombre, descripción y el nivel de la misma. A continuación, el profesor ingresa la información para el primer día de la rutina, comenzando con la duración en minutos de la misma. Luego, mientras el profesor quiera ingresar ejercicios para ese día, el sistema solicita la información de un ejercicio, indicando nombre, descripción, repeticiones, el grupo muscular ejercitado y la cantidad de kilo calorías. Si la rutina tiene más días, se ingresa la información de los días restantes de forma idéntica al primer día de la rutina. Finalmente el profesor confirma y el sistema crea la rutina de ejercicios y la vincula al profesor de la sesión.

Se pide:

a) Modelo de Dominio de la realidad anterior con restricciones en lenguaje natural. Se deben especificar los tipos de los atributos, así como también los datatypes y enumerados utilizados.
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.

<u>Problema 2</u> (30 puntos)

Se cuenta con el siguiente Modelo de Dominio para soportar una funcionalidad de sugerencia de miembros con intereses y especialidades similares en una red social.



Se tiene además el contrato de la siguiente operación del sistema:

Firma	`buscarMatches(idMiembro:Integer):Set(DataMiembro)`
Descripción	Para los miembros comunes se retornan los datos del conjunto de miembros que coincidan en al menos un interés o especialidad del miembro ingresado. Para los miembros premium se retornan los datos del conjunto de miembros que coincidan en al menos un interés o especialidad y en dicha coincidencia tengan al menos 5 puntos en el rating, de otro modo se descarta la coincidencia.
Parámetros	`idMiembro`: Identifica al miembro para el cual se buscarán las coincidencias.
Precondiciones	Existe en el Sistema una instancia de `Miembro` cuyo `id` coincide con `idMiembro`.
Poscondiciones	Si la instancia `m` de `Miembro` con `id` igual a `idMiembro` es de tipo `Comun`, el resultado incluye un datavalue de tipo `DataMiembro` por cada instancia de `Miembro` que esté asociado con al menos una instancia de `Interes` o de `Especialidad` que también esté asociado con `m`. Si `m` es de tipo `Premium`, se considera coincidencia si además el atributo rating de `RatingInteres` o `RatingEspecialidad` es mayor o igual a 5.

Se pide:

a) Realizar un diagrama de comunicación para la operación `buscarMatches`, definiendo todos los datatypes utilizados.
b) Realizar el Diagrama de Clases de Diseño resultante, incluyendo sólo los constructores o destructores que se utilicen en el diagrama de comunicación y excluyendo setters y getters.

Se desea extender las funcionalidades de manera que la operación `buscarMatches` permita a los usuarios del sistema definir en tiempo de ejecución qué algoritmo de coincidencias desean utilizar para cada una de sus búsquedas.

Se pide:

c) Provea un Diagrama de Clases de Diseño anexando tres clases (MatchingA, MatchingB, MatchingC) que implementan tres algoritmos distintos para soportar la funcionalidad diseñada en las partes a y b. Detalle los patrones utilizados, roles de las clases y pseudocódigo en modo de nota de las operaciones principales para explicar su funcionamiento.

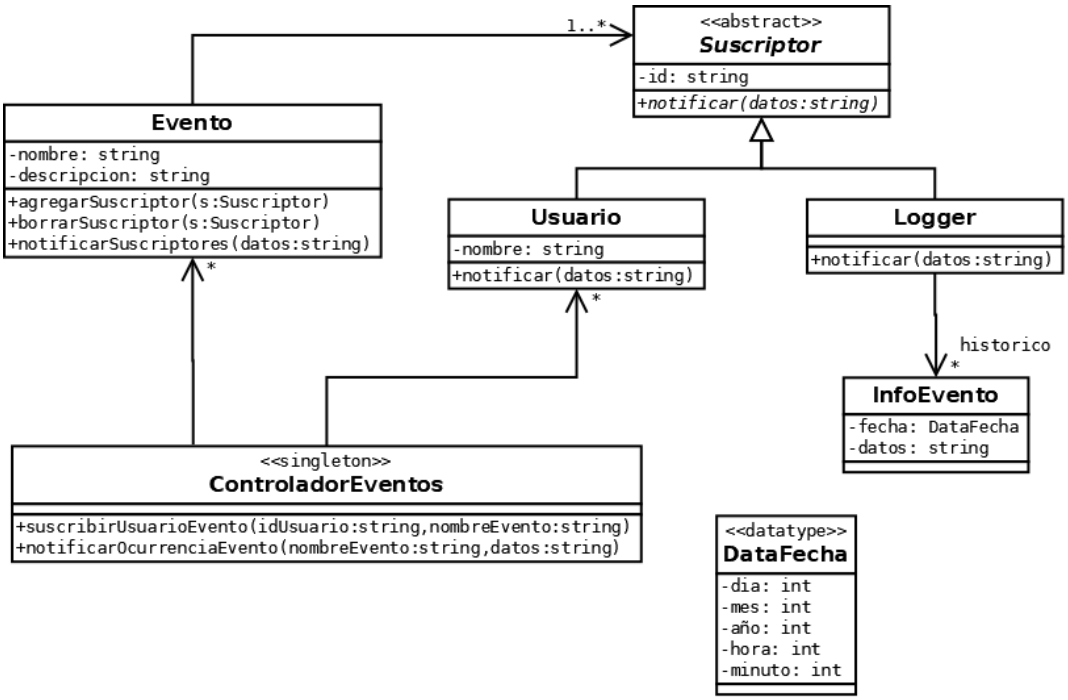
<u>Problema 3</u> (35 puntos)

La figura muestra el Diagrama de Clases de Diseño (DCD) de un software para el registro y notificación de eventos.

El sistema tiene definido distintos eventos a los que pueden suscribirse los usuarios registrados. Cada evento se identifica por un *nombre* y por cada uno de ellos se tiene una instancia de la clase `Evento`. Por otra parte, los usuarios del sistema son identificados por un *id*.

Al ocurrir un evento se invoca (automáticamente y desde afuera del sistema) la operación del controlador `notificarOcurrenciaEvento()`que notificará a todos los usuarios suscriptos a ese evento.

Por último, para llevar el histórico de las ocurrencias de los eventos se tiene una instancia de la clase `Logger` suscripta a cada evento del sistema, cuya función es almacenar la información de los eventos ocurridos.



A continuación se describen algunas de las operaciones.

Clase `ControladorEventos`:

- `suscribirUsuarioEvento()`: suscribe al usuario (identificado por su *id*) al evento (identificado por su *nombre*).
- `notificarOcurrenciaEvento()`: notifica enviando los datos a todos los suscriptos al evento ocurrido.

Clase `Usuario`:

- `notificar():` no hace nada.

Clase `Logger`:

- `notificar():` crea una instancia de `InfoEvento` con los datos del evento y fecha y hora del sistema y la almacena en el histórico. Asuma que existe la operación `Sistema::obtenerFecha():DataFecha` que retorna la fecha y hora actual del sistema.
- No se pide implementar la eliminación del histórico de las ocurrencias de los eventos.

Se pide:

a) Implementar en C++ completamente los *.h* de todas las clases menos el datatype.
b) Implementar en C++ completamente el *.cpp* de todas las clases menos el datatype.

Observaciones:

- Puede utilizar colecciones genéricas (realizaciones de IDictionary e ICollection) o paramétricas (contenedores STL).
- Implementar exclusivamente las operaciones del DCD. Incluir constructores y destructores cuando sea necesario.
- No incluir directivas al precompilador.