EIF204 Programación 2

Proyecto No 1

Prof. Santiago Caamaño P. (Coord.).
Prof. Karol Leitón A.
Prof. Georges E. Alfaro S.
Prof. Juan de Dios Murillo M.

Planteamiento del problema

La aplicación por desarrollar es un sistema simple de gestión para los clientes de un gimnasio. El gimnasio manejará información de sus clientes (triatletas) y les permitirá matricular en diferentes cursos que pueden ofrecer.

Objetivos del proyecto

El proyecto tiene como objetivo estudiar algunas de las técnicas fundamentales de construcción de programas usando Programación Orientada a Objetos (POO) y la aplicación de patrones de diseño, tales como el patrón delegación, utilización del upCasting, entre otros.

Descripción del proyecto

El gimnasio maneja una base de datos sencilla con la información de los clientes que deben ser deportistas de tipo triatlonistas, solamente. A saber, el triatlón clásico se compone de natación, ciclismo y carrera a pie en ese orden. Además, la natación es en aguas abiertas, el ciclismo se realiza en asfalto con bici de carretera y la carrera a pie principalmente en un circuito urbano. Así que un Triatlón de Media Distancia está compuesto por: 3 kilómetros de natación en aguas abiertas. 90 kilómetros de ciclismo. 42.195 metros de carrera a pie.

Hay triatlonistas que han participado en eventos tipo Ironman. Cuando hablamos de IRONMAN lo hacemos de una modalidad de triatlón compuesta por 3.8 kilometros de natación, 180 kilómetros de ciclismo y 42 kilómetros de carrera a pie.

Los clientes

Los clientes tienen un expediente donde, además de los datos personales de cada uno, se registrará el historial de pagos mensuales, para determinar si se encuentran activos o no en el gimnasio.

Los clientes pueden inscribirse en cursos especiales de corta duración. Cada cliente tiene derecho de matricular hasta 4 cursos cortos al mes, dependiendo del cupo disponible. Para ello, el cliente debe reservar su espacio antes de la fecha inicial del curso y mientras exista espacio en los respectivos grupos. Sólo los clientes activos podrán hacer reservaciones.

Para el funcionamiento correcto del sistema y para hacer las pruebas correspondientes, el programa debe solicitar la fecha al iniciar, mostrando la fecha actual por defecto.

La funcionalidad del programa comprende las siguientes operaciones:

- Registro de clientes nuevos
- Registro y control (verificación y reporte) de pagos
- Reportes generales
- Registro y actualización de cursos
- Manejo de reservaciones (incluyendo verificación y cancelación)

Para cada cliente (triatleta), se registrará información básica como:

- Número de identificación (cédula)
- Nombre
- Teléfono
- Fecha de nacimiento
- Sexo
- Datos biométricos básicos: estatura, peso, porcentaje de grasa corporal y de masa muscular. Los datos deben incluir las fechas respectivas de registro.

- **Nota:** Es vital que el programa permita la edición o actualización de la información básica de los clientes (triatletas). Es decir, poder cambiar cualquiera de los datos anteriores.

Los cursos deben incluir:

- Descripción del curso
- Nivel (principiante, intermedio, avanzado)
- Fechas y horario del curso
- Cupo máximo
- Reservas (lista de clientes inscritos)

El programa mostrará en consola algunos reportes básicos, como:

- Lista general de clientes
- Lista de clientes activos (también inactivos o con cuentas pendientes)
- Lista de cursos disponibles
- Lista de cursos con sus respectivos grupos.
- Lista de grupos (clientes que han hecho reservas)

Consideraciones de implementación

Hay que definir una clase manejo simple de fechas, que permita algunas operaciones básicas, como diferencia entre fechas y cálculo de edad.

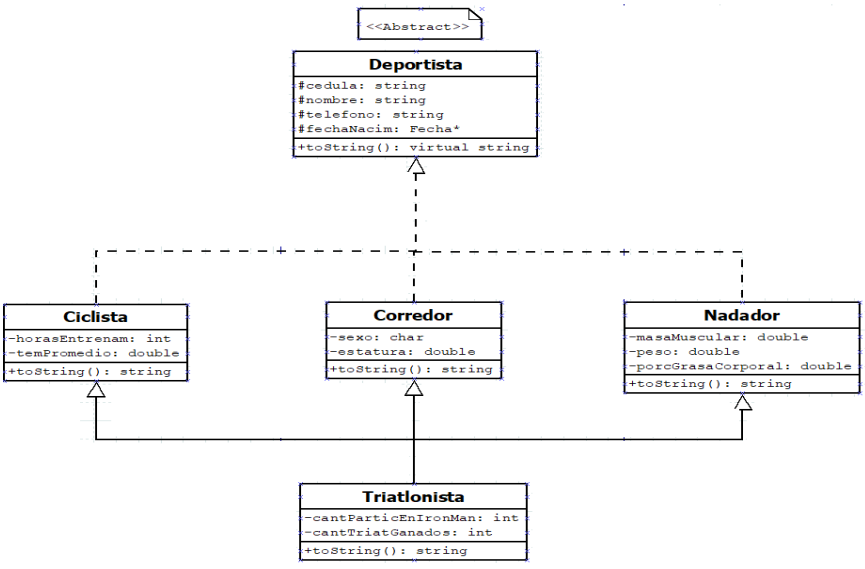
Para el manejo de la interfaz se construirá una clase 'menú' o "interfaz de usuario", que maneje una lista de opciones e incluya métodos para seleccionar entre las opciones disponibles.

Se debe usar una única clase para el manejo de listas, es decir un contenedor genérico (template) que permita el almacenamiento de datos de los clientes, con iteradores para su recorrido. No es permitido el uso de la STL (*Standard Template Library* – biblioteca estándar de plantillas)

Los datos se guardarán en archivos de texto y se recuperarán de la misma manera. Cada vez se ejecute la aplicación se utilizarán los datos ya registrados en el sistema, que se actualizarán durante el uso normal del programa.

Diseñe la arquitectura del programa utilizando UML e incluya los diagramas de clase correspondientes. Si es necesario o si el profesor(a) lo indica explícitamente, incluya otros diagramas, tales como los de secuencia o estado.

En el siguiente esquema se tiene una **herencia múltiple** donde se observa que la clase Triatlonista hereda de tres clases que son: la clase Ciclista, la clase Corredor y la clase Nadador. Con ayuda de algún patrón usted debe transforma esta idea a una que pueda trabajar con mayor facilidad. El nuevo esquema para trabajar deberá ser ingresado en el proyecto. Así que, cuando se crea un cliente triatleta, debe ser de tipo Deportista*. En este esquema se observan los atributos a utilizar, pero los métodos no. Usted debe escribirlos en el esquema nuevo a realizar.



Entrega y evaluación

El proyecto debe entregarse al final de la semana 9 del curso, es decir para el sábado **6 de mayo de 2023 a cualquier hora, antes de las 11:30 pm**. No se aceptará ningún proyecto después de esa fecha. Los trabajos no se copiarán de ninguna memoria USB u otro dispositivo en el momento, sino que se deben entregar en el formato pedido y usando el medio adecuado (el aula virtual institucional o de la manera que expresamente lo indique el profesor).

El proyecto se puede realizar en **grupos de dos o tres personas**, como **máximo**. Recuerde que deberá indicar oportunamente cual es la conformación del grupo de trabajo por medio de un mensaje de correo. Se debe adjuntar un documento con el nombre completo y cédula de cada participante del grupo, indicando el nombre del curso, ciclo lectivo y descripción del trabajo que se entrega.

Incluya comentarios en el código de los programas y describa detalladamente cada una de las clases y métodos utilizados. Los proyectos deben entregarse con diagramas, código fuente y cualquier otro material adicional indicado por el (la) profesor(a).

Se evaluará la funcionalidad del programa y aplicación de los diferentes principios de diseño. La interfaz deberá ser fácil de entender por el usuario, indicando claramente las opciones disponibles.

El programa validará el **formato de los datos de entrada**, para evitar errores de operación por datos incorrectos.

Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta. Cualquier trabajo práctico, que no sea de elaboración original de los estudiantes y haya sido copiado o adaptado de otro origen (plagio), de manera parcial o total, se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.

La selección del IDE a utilizar en la elaboración de los proyectos queda a criterio del (la) profesor(a) del curso. El proyecto se evaluará de acuerdo con la siguiente rúbrica:

Rubros de evaluación	%
Evaluación- 1 ➔ **Patrón Delegate** ➔ Creación de Triatleta	**23%**
Evaluación-2 ➔ **Plantillas-iteradores** ➔ (Templates)	**22%**
Evaluación-3 ➔ **Manejo de Excepciones** ➔ try-throw-catch	**15%**
Evaluación-4 ➔ **Manejo de archivos** ➔ Guardar-Recuperar	**10%**
Evaluación-5 ➔ **Ejecución del proyecto** ➔ Sin errores	**20%**
Evaluación-6 ➔ **Implementación correcta y esquema de un DRC**	**10%**

Nota: Lo más importante de la anterior evaluación (rúbrica) es la Ejecución del proyecto, es decir su funcionalidad, si ese factor falla, los rubros de las otras evaluaciones caerán porque simplemente el código está mal construido y/o mal implementado.

El escribir código no garantiza la nota en los rubros de evaluación, así que, el programa debe correr y ser ejecutado con toda claridad. Por ejemplo: aunque haya código donde se evidencia un "posible trabajo" sobre el manejo de archivos, si el programa no recupera la información después de haber sido ingresada (grabada), la nota en esa evaluación es un cero (0).

Observaciones generales:

Cualquier trabajo práctico que no sea de elaboración original de los estudiantes (plagio) se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.

Referencias

10 OOP Design Principles Every Programmer Should Know. (15 de marzo de 2023). Obtenido de Discover Anything: https://hackernoon.com/10-oop-design-principles-every-programmer-should-know-f187436caf65

Principles of Object-Oriented Design. (15 de marzo de 2023). Obtenido de UTSA CS 3443 Application Programming: http://www.cs.utsa.edu/~cs3443/notes/designPrinciples/designPrinciples.html

SOLID Rules in Object-Oriented Programming. (15 de marzo de 2023). Obtenido de We Are Community: https://wearecommunity.io/communities/epam-poland/articles/1190

SOLID: The First 5 Principles of Object Oriented Design. (15 de marzo de 2023). Obtenido de Digital Ocean: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design