VNIVERSITAT DE VALÈNCIA

Laboratorio de PROGRAMACIÓN
Grado en Ingeniería Multimedia (1º)
Curso 2021-22

Práctica Nº 5: Programación Orientada a Objetos (Herencia)
Periodo de realización: Semana del **04/04/2022** al **08/04/2022**

Material a entregar

- **(Repositorio) Proyecto compilable con el código fuente necesario para la práctica.**
- **(Tarea AV) Documentación del código generada por Doxygen.**

Introducción

En esta práctica se va a utilizar el concepto de herencia entre clases, propio de lenguajes orientados a objetos, para desarrollar una adaptación de la aplicación de la práctica 4 para la organización de eventos de atletismo de alto nivel en los que participarán atletas profesionales a los que se les exigirá una marca mínima para poder ser invitados al evento, tal como ocurre habitualmente en las competiciones oficiales más importantes.

Antes de abordar la aplicación será necesario definir e implementar dos nuevos tipos de datos que permitan representar los dos nuevos conceptos que aparecen en la práctica, el concepto de *atleta profesional* y el concepto de *evento para atletas profesionales*. Estos tipos recibirán los nombres *AtletaProfesional* y *EventoProfesional*, respectivamente. Una vez completados los nuevos tipos (clases), será el momento de desarrollar la aplicación de gestión que haga uso de ellos y cuyo funcionamiento será prácticamente idéntico al de la aplicación de la práctica 4.

Como esta práctica continúa con el trabajo de las prácticas 3 y 4, será necesario recuperar las clases allí utilizadas y cuyo código fuente, obviamente, ya está desarrollado. En particular, será necesario disponer de los siguientes archivos de las prácticas anteriores:

- `Pr3_Atleta.h` y `Pr3_Atleta.cpp`
- `Pr4_Fecha.h` y `Pr4_Fecha.cpp`
- `Pr4_Evento.h` y `Pr4_Evento.cpp`

Ejercicios

SCV: Debes trabajar en la carpeta “Pr5” del repositorio que ya tienes creado. Descarga en ella los 6 archivos indicados en el apartado anterior más los archivos `"separar.h"` y `"atletas_pro_big.csv"`, que se proporcionan como material inicial. Haz un primer *commit* (“Pr.5: Versión inicial”) y *push* con estos archivos.

Tarea 1

La primera clase que se debe desarrollar es la clase `AtletaProfesional`, que derivará/heredará de la clase `Atleta` de la que ya se dispone. Un atleta profesional es un tipo particular de atleta y, por tanto, todas las especificaciones realizadas con carácter genérico para un atleta también son válidas (y reutilizables) para un atleta profesional. El concepto de atleta profesional amplía el de atleta al registrar todos los tiempos realizados en las carreras en las que ha participado. Los cambios en las operaciones están relacionados con la gestión de esta información adicional.

Se debe construir la clase `AtletaProfesional` cumpliendo la siguiente especificación formal:

TAD AtletaProfesional

Dominio: Representa a un atleta profesional, que es un tipo derivado del tipo Atleta. Además de toda la información propia de aquel tipo, un atleta profesional tiene:

- El conjunto de tiempos realizados (marcas) durante la temporada en la especialidad en la que participa. Estos tiempos se registran en segundos, como un número real, con independencia de la especialidad (todas son carreras de diferente distancia).

Operaciones: Todas las operaciones propias de Atleta son válidas y solo hay que considerar las siguientes nuevas operaciones y redefiniciones[1]:

- *`CrearAtletaProfesional()` → `AtletaProfesional`*
- *`IncorporarMarca(AtletaProfesional, Real)` → `AtletaProfesional`*
- *`DevuelveMarcas(AtletaProfesional)` → `Vector de Real`*
- *`DevuelveMejorMarca(AtletaProfesional)` → `Real`*
- *`LeerDeFichero (AtletaProfesional, FlujoEntrada)` → `Lógico`*
- *`Mostrar (AtletaProfesional)`*

Axiomas:

Sea $a \in$ AtletaProfesional, $t \in$ Real, $f \in$ FlujoSalida

`CrearAtletaProfesional ()`	*Crear un atleta profesional. Deberá crear todos los datos propios de un atleta como se crean en ese tipo y el conjunto de tiempos realizados en la temporada estará vacío.*
`IncorporarMarca(a ,t)`	*Añade el tiempo* `t` *al conjunto de tiempos realizados por el atleta* `a`.
`DevuelveMarcas(a)`	*Devuelve (en forma de vector) el conjunto de tiempos realizados por el atleta* `a`.
`DevuelveMejorMarca (a)`	*Devuelve el mejor tiempo realizado por el atleta* `a`.
`LeerDeFichero (a, f)`	*Lee los datos del atleta* `a` *del fichero* `f`. *Devuelve verdadero si se ha podido leer los datos y falso en caso contrario. Redefinición para incluir la lectura de los tiempos realizados en la temporada.*
`Mostrar (a)`	*Muestra toda la información del atleta* `a` *por pantalla. Redefinición para incluir la información de los tiempos realizados en la temporada.*

[1] Una redefinición es una nueva definición de una operación ya existente en la clase base, pero que en la clase derivada se debe introducir alguna variante de funcionamiento.

Comentarios y requisitos para la implementación de la clase:

- La implementación se debe realizar en dos archivos, "`Pr5_AtletaProfesional.h`" y "`Pr5_AtletaProfesional.cpp`".
- Es **obligatorio** derivar la clase `AtletaProfesional` de la clase `Atleta`, por lo que será necesario incluir el archivo "`Pr3_Atleta.h`" (`#include "Pr3_Atleta.h"`) en el interfaz de la nueva clase.
- Los elementos declarados como *`private`* en la clase `Atleta` **no pueden ser** modificados para definirlos como *`protected`*.
- Para almacenar el conjunto de tiempos realizados por el atleta profesional se deberá utilizar la *template* `vector`[2] para definir un vector de números reales (*`vector<float>`*).
- La redefinición de la operación *`LeerFichero`* es necesaria porque ahora una línea del archivo de datos incluirá, además de los datos estándar de cualquier atleta, el listado de tiempos realizados por el atleta profesional que se está leyendo. Esta información irá al final de la línea, después de la especialidad. Los tiempos estarán siempre en segundos y separados por `';'`, como el resto de la información. Sin embargo, hay que tener en cuenta (**muy importante**) que no todos los atletas tendrán el mismo número de tiempos registrados y, por lo tanto, no se sabrá a priori cuántos valores hay que leer (ver archivo `"atletas_pro_big.csv"`[3]). Por ejemplo, estas dos líneas corresponden a dos líneas válidas que aparecen en el archivo de datos:

  ```
  DC749408;ANIES LLEVAT, CHARIF;BRA;100m (H);10.81926435;10.25420235;9.942628353
  AA727034;AKHTER BAJANA, ANGEL RODRIGO;BRA;400m (H);43.17438709
  ```

  El primer atleta tiene registrados 3 tiempos porque ha participado en 3 carreras oficiales, mientras que el segundo solo dispone de 1 valor porque solo ha participado en una carrera.
- Para ayudar a realizar la implementación de la función *`LeerFichero`* se proporciona el archivo "`separar.h`", que incluye 2 funciones ya implementadas que permiten obtener todos los elementos contenidos en un *string* que estén separados por `';'`, sin saber a priori cuántos hay. Lo que devuelve cada función y cuál es la diferencia entre ambas se describe en la documentación incluida en el archivo. Para usar cualquiera de estas funciones solo hay que copiar el archivo en la carpeta de trabajo (si has seguido las instrucciones de este guion ya deberías tenerlo) e incluirlo en el archivo fuente donde se use (`#include "separar.h"`).
- La redefinición de la operación *`Mostrar`* debe escribir en pantalla los datos del atleta profesional, haciendo uso de la operación *`Mostrar`* de la clase *`Atleta`*, y además incluir (como novedad) una nueva línea con todas las marcas registradas para ese atleta, separadas por un espacio en blanco. Por ejemplo:

  ```
  id: KU812080 - AVELLAN UGIA, ANA PAZ
  Nacionalidad: ESP
  Especialidad: 100m (M)
  11.6138 10.603 11.4061 11.4797 11.7428
  ```

Es muy conveniente que realices un pequeño programa de prueba para verificar la clase.

SCV: Registra los cambios de "`Pr5_AtletaProfesional.h`" y "`Pr5_AtletaProfesional.cpp`" (*commit*) con el mensaje "Pr.5: Completada Tarea 1".

[2] Recuerda que para usar este tipo es necesario incluir el archivo de cabecera: `#include <vector>`

[3] NO abras este archivo con Excel. Ábrelo con cualquier otro programa editor de textos que no interprete ni modifique el formato. Por ejemplo, con un bloc de notas o con el mismo Dev-C++.

Tarea 2

La segunda clase que se debe desarrollar es la clase `EventoProfesional`, que derivará de la clase `Evento` de la que ya se dispone. Un evento profesional es un tipo particular de evento deportivo y, por tanto, todas las especificaciones realizadas con carácter genérico para un evento también son válidas (y reutilizables) para un evento profesional. El concepto de evento profesional amplia el de evento al especificar una marca de tiempo mínima que se requerirá a los atletas para poder participar en él. Además, los atletas que participan en este tipo de eventos son siempre atletas profesionales.

Se debe construir la clase `EventoProfesional` cumpliendo la siguiente especificación formal:

TAD EventoProfesional

Dominio: Representa un evento deportivo en el que solo participan atletas profesionales y es un tipo derivado del tipo Evento. Además de toda la información propia de aquel tipo, un evento profesional tiene las siguientes características:

- Los atletas que participan son profesionales.
- Se requiere una marca mínima para participar. De manera que, todos los atletas participantes tienen una mejor marca inferior o igual a la marca mínima del evento.

Operaciones: Todas las operaciones propias de Evento son válidas y solo hay que considerar las siguientes nuevas operaciones y redefiniciones:

- *`CrearEventoProfesional()` → `EventoProfesional`*
- *`DevuelveMarcaMinima(EventoProfesional)` → `Real`*
- *`AsignarValores(EventoProfesional , Cadena, Cadena, Cadena, Cadena, Fecha, Cadena, Real)` → `EventoProfesional`*
- *`InvitarAtleta(EventoProfesional,AtletaProfesional)` → `EventoProfesional`*
- *`GuardarAtletas(FlujoSalida, EventoProfesional)`*
- *`MostrarEvento(FlujoSalida, EventoProfesional)`*

Axiomas:

Sea e ∈ Evento, c1, c2, c3, c4, c5, c6 ∈ Cadena, a ∈ AtletaProfesional, d ∈ Fecha, t ∈ Real, f ∈ FlujoSalida

`CrearEventoProfesional ()`	*Crear un evento profesional. Deberá crear todos los datos propios de un evento como se crean en este tipo y la marca requerida será `0`.*
`DevuelveMarcaMinima (e)`	*Devuelve la marca mínima requerida en el evento `e`.*
`AsignarValores(e, c1, c2, c3, c4, fec, c5, c6, t)`	*Asigna valores a la información del evento e. Redefinición para incluir la marca requerida (`t=marca requerida`): `c1=código`, `c2=nombre`, `c3=ciudad`, `c4=país`, `d=fecha`, `c5=especialidad`, si `c6='Internacional'` el `tipo de evento` será internacional y si no, nacional .*
`InvitarAtleta(e, a)`	*Añade el atleta* a *a la lista de atletas invitados al evento e. Redefinición para incluir atletas profesionales.*
`GuardarAtletas(f, e)`	*Guarda en el **flujo de salida** f los datos de los atletas profesionales invitados al evento e. Redefinición para incluir nueva información de los atletas profesionales.*
`MostrarEvento(f, e)`	*Escribe en el flujo f la información del evento e. Redefinición para incluir la marca requerida.*

Comentarios y requisitos para la implementación de la clase:

- La implementación se debe realizar en dos archivos, "`Pr5_EventoProfesional.h`" y "`Pr5_EventoProfesional.cpp`".
- Es **obligatorio** derivar la clase `EventoProfesional` de la clase `Evento`, por lo que será necesario incluir el archivo "`Pr4_Evento.h`" (`#include "Pr4_Evento.h"`) en el interfaz de la nueva clase.
- Los elementos declarados como *`private`* en la clase `Evento` **no pueden ser** modificados para definirlos como *`protected`*.
- El conjunto de atletas profesionales participantes se representará con la *template* `vector`[4] (*`vector<AtletaProfesional>`*).
- Un evento profesional se construye igual que un evento y adicionalmente solo se debe establecer la marca requerida a `0`.
- La operación *`AsignarValores`* reutilizará la misma operación de la clase *`Evento`* y solo debe asignar además la marca requerida.
- La operación *`GuardarAtletas`* guardará para cada atleta la misma información que la operación equivalente de la clase *`Evento`* (código, nombre, país, especialidad) y además, guardará como último valor la **mejor marca** del atleta.
- El método *`MostrarEvento(f,e)`*, al igual que en la práctica anterior, se debe implementar con la sobrecarga del operador '`<<`' y mostrará la misma información que la operación equivalente de la clase *`Evento`*, pero también mostrará la marca requerida en el evento. Asimismo, mostrará la mejor marca de cada atleta. Ejemplo:

```
############################################################
# 0001 Copa del Mundo (100m (M))
# Valencia (ESP) - 30/05/2022 - (Internacional)
# Marca minima requerida: 11 seg.
############################################################

Participantes invitados
------------------------------------------------------------
ALLOUCH OYAGUE, ALYSON (POR) 10.5185 seg.
ARCINIEGAS GALLARDO, ANALIA (ARG) 10.8332 seg.
ARCOS CARPENTE, BEATRIZ PATRICIA (ITA) 10.7998 seg.
AROCENA MAHJOUBI, ANA MARIA JESUS (ITA) 10.5512 seg.
AVELLAN UGIA, ANA PAZ (ESP) 10.603 seg.
BALLESTE MARGALLO, BEATRIZ INMACULADA (SUE) 10.9562 seg.
BATLLO LLORIS, CIRCUNCISION (BRA) 10.7021 seg.
BORNAS FORNAS, ADRIANA MARIA (POR) 10.9424 seg.
```

Es muy conveniente que realices un pequeño programa de prueba para verificar que las operaciones de la clase que has implementado funcionan correctamente.

SCV: Registra los cambios de "`Pr5_EventoProfesional.h`" y "`Pr5_EventoProfesional.cpp`" (*commit*) con el mensaje "Pr.5: Completada Tarea 2".

[4] Recuerda que para usar este tipo es necesario incluir el archivo de cabecera: `#include <vector>`

Tarea 3

Por último, se debe realizar un programa que gestione eventos profesionales, tal como se hizo en la práctica 4. Se debe crear un nuevo proyecto en Dev-C++ (“`Pr5.dev`”) al que debes añadir todos los archivos correspondientes a las clases utilizadas (*Atleta*, *AtletaProfesional*, *Evento*, *EventoProfesional*, *Fecha*) y un nuevo fichero “`Pr5.cpp`” para el programa principal. Las tareas que debe realizar el programa son las mismas que el programa de la práctica 4 (puedes reutilizarlo[5] y adaptarlo a las nuevas clases):

- Leer y cargar en memoria (en forma de `vector`[6] de atletas profesionales) el listado de deportistas disponible en el fichero "`atletas_pro_big.csv`".
- Iniciar correctamente un nuevo evento profesional, solicitando la información necesaria (incluida la marca mínima requerida).
- Invitar a participar en el evento a los atletas profesionales leídos inicialmente que cumplan con las condiciones para participar, es decir, que:
    - Tengan asignada la especialidad del evento.
    - Sean del país en el que se celebra la competición si el evento es nacional o de cualquier país si el evento es internacional.
    - Su mejor marca personal sea menor o igual que la marca mínima requerida en el evento.
- Escribir en un fichero toda la información del evento. Al igual que en la práctica 4, la salida por fichero debería ser muy sencilla si se ha sobrecargado correctamente el operador '<<', ("`f << e`", siendo `f` un fichero de tipo `ofstream` correctamente abierto y `e` un evento profesional). El nombre del archivo de salida será el código del evento y su extensión '`.info`'.
- Generar un fichero solo con la información de los atletas en formato '`.csv`' separado por '`;`', tal y como teníamos la información en el fichero original, pero incluyendo solo la mejor marca de cada atleta participante (no todas las marcas). El nombre del archivo de salida será el código del evento y su extensión '`.csv`'.

Para validar el programa utiliza el siguiente **banco de pruebas**:

(continúa en la siguiente página …)

[5] Respecto a la práctica 4 esta tarea debe requerir muy pocas modificaciones. Tan solo algún cambio en la especificación de los tipos utilizados y una condición adicional al invitar a los atletas. Poco más.

[6] *Template* estándar de C++. Este es un cambio que simplifica el programa de la práctica 4. Ten en cuenta que el número de elementos contenidos en un vector *v* lo indica el método *`v.size()`*.

Prueba 1

```
Datos del evento:
        Código: 001
        Nombre: Copa de Valencia
        Ciudad: Valencia
        País: ESP
        Fecha: 20-05-2022
        Especialidad: 100m (H)
        Internacional (S/N)? N
        Marca mínima requerida para participar (segs.): 10

Salida por pantalla:
        ################################################################
        # 001 Copa de Valencia (100m (H))
        # Valencia (ESP) - 20/05/2022 - (Nacional)
        # Marca minima requerida: 10 seg.
        ################################################################

        Participantes invitados
        ----------------------------------------------------------------
        DUGO GONDA, BENJAMIN JAMES (ESP) 9.75424 seg.
        INCA ALLEPUZ, COSTELA (ESP) 9.65797 seg.
        SABUGUEIRO BOU, DIEGO GERARDO (ESP) 9.64584 seg.
        ----------------------------------------------------------------
        El evento ha sido guardado de forma correcta.
        Los atletas han sido guardados de forma correcta.

Contenido del archivo “001.info”:
        ################################################################
        # 001 Copa de Valencia (100m (H))
        # Valencia (ESP) - 20/05/2022 - (Nacional)
        # Marca minima requerida: 11.5 seg.
        ################################################################

        Participantes invitados
        ----------------------------------------------------------------
        DUGO GONDA, BENJAMIN JAMES (ESP) 9.75424 seg.
        INCA ALLEPUZ, COSTELA (ESP) 9.65797 seg.
        SABUGUEIRO BOU, DIEGO GERARDO (ESP) 9.64584 seg.
        ----------------------------------------------------------------

Contenido del archivo “001.csv”:
        TL578053;DUGO GONDA, BENJAMIN JAMES;ESP;100m (H);9.75424
        NB547664;INCA ALLEPUZ, COSTELA;ESP;100m (H);9.65797
        NK772532;SABUGUEIRO BOU, DIEGO GERARDO;ESP;100m (H);9.64584
```

(Prueba 2 en la siguiente página …)

Prueba 2

```
Datos del evento:
        Código: 002
        Nombre: Copa del Mundo
        Ciudad: Zurich
        País: SUI
        Fecha: 01-06-2022
        Especialidad: 10000m (M)
        Internacional (S/N)? S
        Marca mínima requerida para participar (segs.): 1800

Salida por pantalla:
        ################################################################
        # 002 Copa del Mundo (10000m (M))
        # Zurich (SUI) - 01/06/2022 - (Internacional)
        # Marca minima requerida: 1800 seg.
        ################################################################

        Participantes invitados
        ----------------------------------------------------------------
        AYOUBI DORESTE, CLIO (AUS) 1796.34 seg.
        BARCO PENAFIEL, ANA PETRA (SUI) 1790.8 seg.
        BELKHIR MONTERRUBIO, DENICE (DIN) 1783.74 seg.
        BEVAN MELLOUK, AITANA MARIA (GBR) 1779.69 seg.
        BRIME KRAFT, ADRIANA VANESSA (HON) 1783.34 seg.
        CAAVEIRO SEBTI, ALBERTINA MARIA (BRA) 1778.23 seg.
        CASBAS LAURA, DOLORES M (ESP) 1799.08 seg.
        CEJAS CORMENZANA, ANA PIEDAD (FRA) 1794.44 seg.
        DABRIO USATEGUI, CRISTINA PAOLA (ARG) 1761.06 seg.
        GRANO DE ORO BURCHES, AGLAE (ESP) 1749.11 seg.
        MONTENEGRO VOICA, CLEMENTA (GBR) 1770.9 seg.
        SANCHEZ CABEZUDO ALCEGA, DARA MARIA (GBR) 1794.16 seg.
        SANGUINETTI GASCH, ANA RALUCA (HOL) 1772.23 seg.
        ----------------------------------------------------------------
        El evento ha sido guardado de forma correcta.
        Los atletas han sido guardados de forma correcta.

Contenido del archivo “002.info”:
        ################################################################
        # 002 Copa del Mundo (10000m (M))
        # Zurich (SUI) - 01/06/2022 - (Internacional)
        # Marca minima requerida: 1800 seg.
        ################################################################

        Participantes invitados
        ----------------------------------------------------------------
        AYOUBI DORESTE, CLIO (AUS) 1796.34 seg.
        BARCO PENAFIEL, ANA PETRA (SUI) 1790.8 seg.
        BELKHIR MONTERRUBIO, DENICE (DIN) 1783.74 seg.
        BEVAN MELLOUK, AITANA MARIA (GBR) 1779.69 seg.
        BRIME KRAFT, ADRIANA VANESSA (HON) 1783.34 seg.
        CAAVEIRO SEBTI, ALBERTINA MARIA (BRA) 1778.23 seg.
        CASBAS LAURA, DOLORES M (ESP) 1799.08 seg.
        CEJAS CORMENZANA, ANA PIEDAD (FRA) 1794.44 seg.
        DABRIO USATEGUI, CRISTINA PAOLA (ARG) 1761.06 seg.
        GRANO DE ORO BURCHES, AGLAE (ESP) 1749.11 seg.
        MONTENEGRO VOICA, CLEMENTA (GBR) 1770.9 seg.
        SANCHEZ CABEZUDO ALCEGA, DARA MARIA (GBR) 1794.16 seg.
        SANGUINETTI GASCH, ANA RALUCA (HOL) 1772.23 seg.
        ----------------------------------------------------------------
```

(continúa en la siguiente página …)

Contenido del archivo "002.csv":

```
GJ765833;AYOUBI DORESTE, CLIO;AUS;10000m (M);1796.34
HK138292;BARCO PENAFIEL, ANA PETRA;SUI;10000m (M);1790.8
HG660839;BELKHIR MONTERRUBIO, DENICE;DIN;10000m (M);1783.74
FZ804912;BEVAN MELLOUK, AITANA MARIA;GBR;10000m (M);1779.69
RH426593;BRIME KRAFT, ADRIANA VANESSA;HON;10000m (M);1783.34
WU151103;CAAVEIRO SEBTI, ALBERTINA MARIA;BRA;10000m (M);1778.23
YY844045;CASBAS LAURA, DOLORES M;ESP;10000m (M);1799.08
G82005;CEJAS CORMENZANA, ANA PIEDAD;FRA;10000m (M);1794.44
LQ837558;DABRIO USATEGUI, CRISTINA PAOLA;ARG;10000m (M);1761.06
JI518183;GRANO DE ORO BURCHES, AGLAE;ESP;10000m (M);1749.11
MX213759;MONTENEGRO VOICA, CLEMENTA;GBR;10000m (M);1770.9
TU980668;SANCHEZ CABEZUDO ALCEGA, DARA MARIA;GBR;10000m (M);1794.16
YS671261;SANGUINETTI GASCH, ANA RALUCA;HOL;10000m (M);1772.23
```

SCV: Registra los cambios realizados en los archivos del proyecto (*commit*) con el mensaje "Pr.5: Versión final, completada Tarea 3". Haz *push*.

Tarea 4: Generación de documentación

El código realizado deberá estar correctamente documentado con Doxygen, de acuerdo con la guía de estilo. Una vez realizada la versión final del programa, se deberá ejecutar Doxygen para generar toda la documentación definitiva. La documentación deberá ser generada exclusivamente en formato en *html* (utilizar como referencia, convenientemente personalizado para esta práctica, el archivo de configuración de Doxygen disponible en el Aula Virtual). Solo la documentación deberá subirse a Aula Virtual (como archivo comprimido). **El programa NO se debe subir al AV, ya que será evaluado directamente desde el repositorio de control de versiones**.

La documentación generada con Doxygen NO debe subirse al repositorio de control de versiones Se debe mantener exclusivamente en el ordenador personal.