PROGRAMACIÓN ORIENTADA A OBJETOS
Parcial N°1

Docente: Víctor Saldivia Vera - Email: victor.saldivia@ulagos.cl
Ingeniería Civil en Informática - Departamento de Ciencias de la Ingeniería

UNIVERSIDAD DE LOS LAGOS

Jueves 16 de Octubre de 2025

Enunciado
Resolver el siguiente ejercicio utilizando únicamente los conocimientos adquiridos en la Unidad I y Unidad II de POO. No se permite el uso de código ni material extra durante el Laboratorio. Se solicita desarrollar la solución en Python. Tiempo para resolver el laboratorio: 90 minutos. Puntaje total: 100 puntos.

Observaciones
- Cualquier intento de copia será evaluado con la nota mínima.
- Prohibido el uso de dispositivos móviles durante el control.

1. Implementar Clase Jugador

El club Deportes Castro necesita modelar su plantel: datos básicos, estado físico (energía) y rendimiento (goles). Se requiere una clase robusta que asegure encapsulación, invariantes y uso correcto de decoradores.

A. Constructor e Invariantes (20 pts)
Implementa la clase Jugador con atributos privados:

I. **Atributos Privados:** *nombre*, *edad*, *posicion* (en posición utiliza valores como "DEL", "VOL", "DEF", "ARQ"). El atributo *goles* inicia en 0.
II. **Atributos Públicos:** *club* es un str no vacía y *energia* en [0, 100]
III. Constructor *(__init__)*: Válida entradas, tanto no vacíos como rangos válidos y deja las siguientes invariantes de clases:
   - *edad* >= 15
   - *posicion* pertenece a una posición válida.
   - 0 <= *energia* <= 100
   - *goles* >= 0
   - *club* no vacío
   - Al final del *__init__*, incluye un assert que verifiquen todas las invariantes anteriores.

B. Getters y Setters (20 pts)
Implementa propiedades solo para los **atributos privados**:
- *nombre*: getter (setter opcional: no vacío, re-verificar invariantes con assert).
- *edad*: getter + setter (Precondición: valor >= 15; Postcondición: invariantes; assert).
- *posicion*: getter (setter opcional: usa *posicion_valida*, assert).
- *goles*: solo lectura (sin setter).

Nota sobre los atributos públicos:

- *club* y *energia* pueden ser modificados directamente (por ejemplo: *jugador.club = "Deportes Castro"*; *jugador.energia = 85*).
- **Observación:** toda modificación directa a estos campos debe respetar Precondición y Postcondición e invariantes. En tu código, cada método que cambie estado (por ejemplo: *entrenar()*) debe dejar *energia* en [0,100] y rematar con assert. En tus pruebas, si decides asignar directamente, documenta la precondición y verifica después con un assert (por ejemplo: assert 0 <= *jugador.energia* <= 100 y assert *jugador.club* != "").

C, Métodos de Clase (15 pts)

I. *entrenar(minutos)*
   - **Precondición:** minutos > 0.
   - **Efecto:** reduce *energia* con una regla simple (por ejemplo: *por cada minuto de entrenamiento, baja 1 punto de energía*).
   - **Postcondición:** 0 <= *energia* <= 100.
   - Finaliza con assert de invariantes.

II. *anotar_gol()*
   - Incrementa *goles* en 1.
   - **Postcondición:** *goles* >= 0.
   - Finaliza con assert de invariantes.

D. Métodos mágicos (15 pts)

I. *__str__()*: representación del nombre, club (público), posición, energía (pública), goles.

E. Variable de Clase y Método Estático (15 pts)

I. **Variable y método de clase (Global)**
   A. JUGADORES_CREADOS = 0 (variable de clase, contador global que incrementa en *__init__*).
   B. *@classmethod creados()* debe retornar cuántos jugadores se han creado hasta ahora (contador global).

II. **Método estático**
   A. *@staticmethod posicion_valida(valor)* debe devolver un True o False según si la posición del jugador es válida (úsalo en constructor y/o setters).

2. Hacer Pruebas Mínimas (15 pts)

- Crea 2 jugadores de Deportes Castro.
- Modifica directamente *club* y *energia* y, tras cada cambio, verifica con assert que las invariantes se mantienen (ejemplo assert *jugador.club* != "", assert 0 <= *jugador.energia* <= 100).
- Llama a *entrenar()* y *anotar_gol()*; muestra jugadores ordenados con el método **sorted()**
- Muestra el contador global: *Jugador.creados()* y el estado del jugador con *print(jugador)*.

Instrucciones Generales

- Crear un nuevo archivo Python **(.py)** para el ejercicio.
- Ejecutar el archivo y asegurarse de que los resultados mostrados en pantalla sean correctos.
- Comentar el código para explicar cada paso y operación realizada.
- Subir el archivo de Python **(.py)** en la plataforma ULagos Virtual.
- Este código debe estar en su repositorio personal de GitHub/GitLab.