Examen de Programación Orientada a Objetos

11/08/2025

Contexto

La organización de eSports "*PowerLeague"* ha decidido informatizar su sistema de inscripción y seguimiento de jugadores que participan en sus torneos. El objetivo es gestionar de forma eficiente la información de los distintos tipos de jugadores, los cuales pueden ser: **Jugadores individuales**, que compiten en modalidad 1 contra 1 y **Jugadores de equipo**, que forman parte de partidas por escuadra.

De todos los Jugadores se necesita registrar: nickname, juego y nivel (principiante, intermedio o profesional).
Además, todos los jugadores pertenecen al mismo país de origen, "Argentina", por lo que este dato debe ser representado mediante una variable de clase en la clase Jugador.

De un jugador individual se registra, además: modalidad de control preferida (teclado, joystick, etc.), cantidad de torneos ganados y medida del rendimiento (reacciones por minuto).
El puntaje total se calcula como:

- puntaje = base_por_nivel + (torneos_ganados * 50) + (reacciones_por_minuto * 2)

Donde base_por_nivel es:
- Principiante: 100 puntos
- Intermedio: 200 puntos
- Profesional: 300 puntos

De un jugador de equipo se registra, además: su historial de partidas jugadas y rol dentro de equipo (soporte, líder, atacante, defensa).
El puntaje total se calcula como:

- puntaje= base_por_nivel + 50 puntos por cada victoria + (1 punto por cada minuto jugado) + bonificación_por_rol.

Donde bonificación_por_rol es:
- Líder: +100 puntos
- Atacante: +75 puntos
- Defensor: +60 puntos
- Soporte: +50 puntos

De cada partida que forma parte del historial, se necesita registrar la siguiente información: nombre del equipo rival, resultado (victoria o derrota), y duración en minutos.
Estas partidas son propias del jugador: si se elimina el jugador, también se elimina su historial.

Reglas de negocio: El historial de partidas está compuesto exclusivamente por el jugador de equipo. No se permite compartir objetos Partida entre jugadores distintos.

El analista le solicita a usted que desarrolle una aplicación con las siguientes restricciones:

a) Definir las clases de la jerarquía, con los métodos y atributos correspondientes a cada clase de la narrativa planteada. Y una clase **Gestor de Jugadores**, utilizando una **lista definida por el programador** para almacenar los distintos tipos de jugadores.
b) Leer los datos de jugadores desde el archivo "Jugadores.csv", y agregarlos al principio de la lista. El primer carácter de cada línea indica el tipo de jugador: 'I' para individual, 'E'

para equipo. En el caso de los jugadores de equipo, sus partidas se leen desde el archivo “Partidas.csv” (relacionadas por nickname). Ambos archivos tienen como separador “;”.

c) Implementar un programa principal con un menú de opciones que permita testear las siguientes funcionalidades:

1. Leer por teclado un valor de puntaje mínimo, y mostrar el nickname de los jugadores cuyo puntaje total sea igual o superior a dicho valor.
2. Mostrar para todos los jugadores: nickname, tipo de jugador y su puntaje total.
3. Leer por teclado el nickname de un jugador, si no lo encuentra en el gestor, lanzar la excepción ValueError, y controlarla en donde corresponda, si lo encuentra, si es un jugador de equipo, mostrar el rol de equipo, el historial de partidas jugadas y el puntaje obtenido.
4. Leer por teclado los datos de un nuevo jugador individual, y una posición en la lista. Si la posición es válida (mayor o igual que 0 y menor o igual a la cantidad de elementos), y el nickname es válido, insertar el jugador en la posición indicada. Realizar las siguientes validaciones:
   a) Si la posición no es válida, el método del Gestor de Jugadores debe lanzar una excepción IndexError. La excepción debe ser capturada en el programa principal y debe mostrar el mensaje:
      "La posición ingresada no es válida. No se pudo insertar el jugador."
   b) El nickname no debe repetirse, si se repite no se inserta el jugador individual en la lista, lanzando la excepción ValueError. La excepción debe ser capturada en el programa principal y debe mostrar el mensaje:
      “Nickname repetido, intente con otro nickname”.

Reglas de Negocio: Se debe aplicar polimorfismo de subclase para maximizar la reutilización de código (**NO SE DEBE REPETIR EN LAS SUCBLASES NI UNA LÍNEA DE CÓDIGO**). No se deben crear sublistas por tipo de jugador dentro del gestor.

Regla de Negocio: el nickname es único, no puede repetirse.

Reglas del Analista: NO SE PERMITEN BUSCAR DATOS (DATOS QUE SON ÚNICOS) EN LA LISTA CON EL BUCLE FOR.