Equipos de eSports

En un **torneo internacional de videojuegos competitivos**, cada equipo de eSports está formado por distintos jugadores que desempeñan roles específicos dentro del juego. Cada equipo se identifica por su **nombre**, su **país** y la **suma total de puntos** obtenidos por sus jugadores.

Clases y estructura del programa

Clase `Equipo`

- Atributos:
  - `nombre` (String): nombre del equipo.
  - `pais` (String): país de origen.
  - `totalPuntos` (tipo int): representa la suma total de los puntos conseguidos por los jugadores del equipo.
  - Una lista que almacena a los jugadores inscritos en el equipo.
- Los atributos deben ser **privados**.
- Métodos protegidos:
  - **Getters y setters** para cada atributo.
  - `imprimirDatosEquipo()` : imprime el nombre, país y total de puntos del equipo.
  - `nuevoJugador()` : añade un nuevo jugador al equipo.
  - `calcularTotalPuntos()` : actualiza el atributo `totalPuntos` sumando los puntos de todos los jugadores del equipo.
  - `listarJugadores()` : muestra los nombres (o alias) de todos los jugadores que forman parte del equipo junto con la puntuación de cada uno
  - `buscarJugadorPorId(int id)` : busca y muestra los datos de un jugador según su identificador. Si no existe, se mostrará un mensaje apropiado.

Clase abstracta `Jugador`

Representa a un jugador profesional de eSports.

- Atributos (privados):
  - `id` (int): identificador único del jugador.
  - `alias` (String): nombre o apodo usado en el juego.
  - `puntos` (int): puntuación acumulada en el torneo.
- Métodos:
  - Constructor que inicializa los atributos.

- Getters y setters para cada atributo.
- Equals por id
- `addPuntos(int puntos)` : incrementa los puntos obtenidos por el jugador en una partida.
- `toString()` : imprime los datos básicos del jugador.
- Método abstracto `imprimirRol()` : devuelve un `String` con el rol o tipo de jugador (según su especialización).

Subclases de `Jugador`

Clase `Tanque`

Representa un jugador especializado en absorber daño y proteger al equipo.

- Nuevos atributos (privados):
  - `defensaPromedio` (double): cantidad media de daño bloqueado por partida.
  - `dañoRecibido` (double): daño promedio recibido por partida.
- Métodos:
  - Constructor que llama al de la clase padre.
  - Getters y setters.
  - `imprimirDatos()` : imprime los datos comunes y los específicos del rol.
  - `imprimirRol()` : devuelve “Tanque”.

Clase `Apoyo`

Representa al jugador encargado de curar y asistir al resto del equipo.

- Nuevos atributos (privados):
  - `curacionesPorPartida` (int): número medio de curaciones otorgadas en cada partida.
  - `eficienciaAsistencia` (float): eficacia global del jugador en asistencias (porcentaje).
- Métodos:
  - Constructor que llama al de la clase padre.
  - Getters y setters.
  - `toString()` : imprime todos los datos heredados y específicos.
  - `imprimirRol()` : devuelve “Apoyo”.

Clase `Asesino`

Representa al jugador ofensivo, centrado en infligir daño y eliminar oponentes.

- Nuevos atributos (privados):
  - `precisionPromedio` (float): nivel medio de acierto en ataques.
  - `bajasPorPartida` (int): número promedio de eliminaciones por encuentro.
- Métodos:
  - Constructor que llama al de la clase padre.
  - Getters y setters.
  - `toString()` : imprime los datos del jugador y los particulares del rol.
  - `imprimirRol()` : devuelve “Asesino”.

Clase de prueba (`Main`)

En el método `main` , se debe:

1. Crear un equipo.
2. Añadir jugadores de diferentes tipos (Tanque, Apoyo, Asesino).
3. Permitir al usuario interactuar mediante un menú con las siguientes opciones:

```
- Imprimir datos del equipo
- Añadir un jugador (se piden sus datos)
- Añadir puntos a un jugador tras una partida (id, puntos)
- Calcular el total de puntos del equipo
- Listar todos los jugadores
- Buscar un jugador (por identificador)
- Salir
```
