NOMBRE Y APELLIDOS		ETAPA / CICLO / MÓDULO	CURSO	CONVOCATORIA
		CFGS DAW/DAM - PROGRAMACIÓN	1º	CONTINUA
DNI	FECHA	NOTA		
	10/03/2025			

Imagina que estás desarrollando un **sistema de gestión de partidos deportivos** maneja diferentes tipos de campeonatos y disciplinas, como baloncesto, futbol, etc...

Cada equipo que participa tiene características específicas, como un nombre y un conjunto de jugadores. Además, también existen diferentes tipos de jugadores con sus características propias del deporte que practiquen.

Importante:

- **Implementa los getters de los atributos, pero solo los setter necesarios por el enunciado.**
- **Se recomienda separar las diferentes clases en folios distintos, ya que durante el examen puede darse el caso de modificar alguna clase.**
- **No desarrolles ninguna clase principal o main**

Requisitos generales:

1. **(1,50p)** Todos los **Equipos** sean del deporte que sean, estarán formados por un nombre y un conjunto de jugadores. Cualquier equipo sea del deporte que sea deberá ser capaz de:
    a. Añadir un jugador
    b. Proporcionar el nombre del equipo
    c. Proporcionar un listado de jugadores ordenados por su orden natural.

2. **(0.75p)** Para poder implementar el campeonato de baloncesto, se necesita también definir un **Equipo de Baloncesto**. El equipo de baloncesto tendrá todas las características y funcionalidades de un equipo más la posibilidad de listar sus jugadores ordenados por la posición de juego. A misma posición, por orden alfabético.

3. **(0.75p)** Los **Jugadores**, tendrán un nombre y una posición.

4. **(0,50p)** La **Posición**, es simplemente una enumeración de posiciones: BASE, ALERO y PIVOT. Las enumeraciones deben ser capaces de almacenar su texto con la primera letra en mayúscula y el resto en minúscula.

5. **(3,5p)** Para comenzar el desarrollo de aplicación, el cliente quiere comenzar con una prueba con el baloncesto. Para ello deberás crear un **Campeonato de Baloncesto**. Un campeonato de baloncesto estará formado por:
    i. Un nombre.
    ii. Un conjunto de equipos.
    iii. Y un listado con los partidos celebrados.

    Además, se puede:

    i. Agregar un equipo al campeonato.
    ii. Agregar un partido al campeonato.
    iii. (**1p**) Mostrar la clasificación. (Anexo 1)
    iv. (**1p**) Obtener ganador/es (Anexo 2)
    v. El campeonato, para que se pueda integrar en la aplicación, debe ser **Jugable**

6. (**0,5p**). **Jugable** es una interfaz, que cualquier campeonato que se quiera añadir en el futuro deberá cumplir. Esta interfaz estará definida por las siguientes condiciones:
    i. Que se pueda añadir un equipo, no devolverá nada
    ii. Que se pueda agregar un partido, no devolverá nada
    iii. Que se pueda mostrar una clasificación, devolverá un String con la clasificación
    iv. Que muestre el ganador, devolverá una lista con el/los equipos ganadores

7. **(1p)** Para cada **Partido** se debe almacenar:
    a. El equipo que juega como local
    b. El equipo que juega como visitante
    c. Puntos anotados por el local.
    d. Puntos anotados por el visitante.

    Además, debe tener las siguientes funcionalidades/restricciones:
    e. Cuando se crea un partido, se debe hacer a partir de asignar los equipos local y visitante. El marcador debe ser 0 para ambos equipos en el momento de la creación.
    f. Se debe poder establecer el marcador de ambos equipos, proporcionando los puntos del equipo local y los puntos del equipo visitante, pero con la condición de que no sea empate. Si el marcador es empate o se asignan puntuaciones negativas se deberá lanzar una excepción con un mensaje significativo.
    g. Se debe poder devolver el equipo ganador de un partido, y también se debe poder devolver el equipo perdedor de un partido.

8. **(1,5p)** Finalmente tendremos una clase **GestorFicheros** que tendrá únicamente tres métodos estáticos:
    a. Un método para guardar cualquier objeto Jugable en un fichero. El nombre del fichero se le deberá pasar también al método.
    b. Otro método que devuelva un objeto Jugable a partir del nombre de un fichero.
    c. Un tercer método capaz de guardar en un fichero de texto los datos de un campeonato. Donde cada línea estará formada por:

    [nombre_campeonato];[nombre_equipo];[jugador1]; .... ;[jugadorN]

    Ejemplo del fichero CSV generado:

```
Autonomico;Eliana BC;Andrea R.;Blanca R.;Davinia L.;Luna C.;Pepa C.
Autonomico;Pobla BC;Antonio D.;Dionisio M.;Manuel P.;Miquel J.;Pepe G.
Autonomico;Lliria BC;Carmelo G.;Joaquín S.;Josep R.;Nancho N.;Vicent P.
```

Anexo 1.

```
Clasificación del Campeonato: Autonómico
Equipo            PT    PJ    PG    PP
Llíria BC         9     6     3     3
Pobla BC          9     6     3     3
Eliana BC         6     6     0     6
```

Para cada partido ganado (PG) se suman 2 puntos.

Para cada partido perdido (PP) se suma 1.

El orden de la clasificación es por máxima cantidad de puntos (PT).

Si dos equipos empatan en puntos, se ordena por partidos ganados (PG).

Si dos equipos empatan también en partidos ganados, se ordena por orden alfabético.

IMPORTANTE

Para realizar este apartado, puedes crear las clases que necesites, así como añadir las funcionalidades que consideres a las cualquiera de las clases previamente creadas.

Anexo 2

Según la clasificación del anexo 1, podemos observar que existen dos equipos empatados a puntos. Para establecer el ganador/es, es únicamente por puntos, por lo que los ganadores serían:

Lliria
Pobla

IMPORTANTE

Para realizar este apartado, puedes crear las clases que necesites, así como añadir las funcionalidades que consideres a las cualquiera de las clases previamente creadas.

Ayuda 1

```
public static final Comparator<Object> EJEMPLO_CON_CLASE_OBJECT =
new Comparator<Object>() {
    @Override
    public int compare(Object o1, Object o2) {
        return 0;
    }
};
```