1 Práctica de Programación. Liga Profesional de Futbolín

1.1 Objetivos

Se pide implementar una aplicación para la gestión de Campeonatos Deportivos. La idea es que la aplicación pueda extenderse de forma sencilla para añadir nuevos campeonatos. Por ese motivo el diseño de la aplicación debe estar formado por dos partes: un conjunto de clases comunes a cualquier campeonato (para ello se recomienda el uso de clases genéricas, herencia, interfaces, clases abstractas), y por otra la implementación del campeonato seleccionado: Futbolín y Petanca. Para su desarrollo el estudiante deberá usar todas las herramientas vistas durante la asignatura y que mejor se adapten al desarrollo de la aplicación.

1.2 Introducción al sistema

Nuestra empresa ha decidido cambiar el sistema informático que utiliza para gestionar distintos campeonatos profesionales. Además de la gestión del personal, equipos, ligas, temporadas, etc., el sistema se utilizará como base para simular partidos y, por tanto, el campeonato, utilizando para ello unos coeficientes para jugadores y equipos. Después de varias entrevistas, el modelo que mejor recoge las pretensiones de la gestión de campeonatos es el siguiente:

- El campeonato se encuentra formado por la colección de todas las temporadas que se han jugado, que se identifica con un nombre. Estas temporadas a su vez se encuentran formadas por las distintas ligas que tiene. Por cada liga, se debe gestionar sus jornadas y partidos. En nuestros campeonatos consideraremos que cada partido está formado por un conjunto de enfrentamientos indeterminados: en el caso de futbolín serán de dos jugadores de cada equipo y en el caso de la petanca de uno.
- El campeonato también se encuentra formado por los equipos, los cuales tienen un nombre identificativo, así como su pertenencia a una determinada ciudad y el directivo. El campeonato se encuentra formado por todas las personas que se encuentran en él: Directivos, jueces y jugadores. Los detalles de cada clase se proporciona en el diagrama de clase.

1.3 Análisis del Juego

Las funciones que deben implementarse se muestran en el diagrama de caso de uso de la figura 1.1 (cualquier cambio se notificará en el foro de la asignatura). El enunciado de esta entrega se especificará en la entrega del campusvirtual.

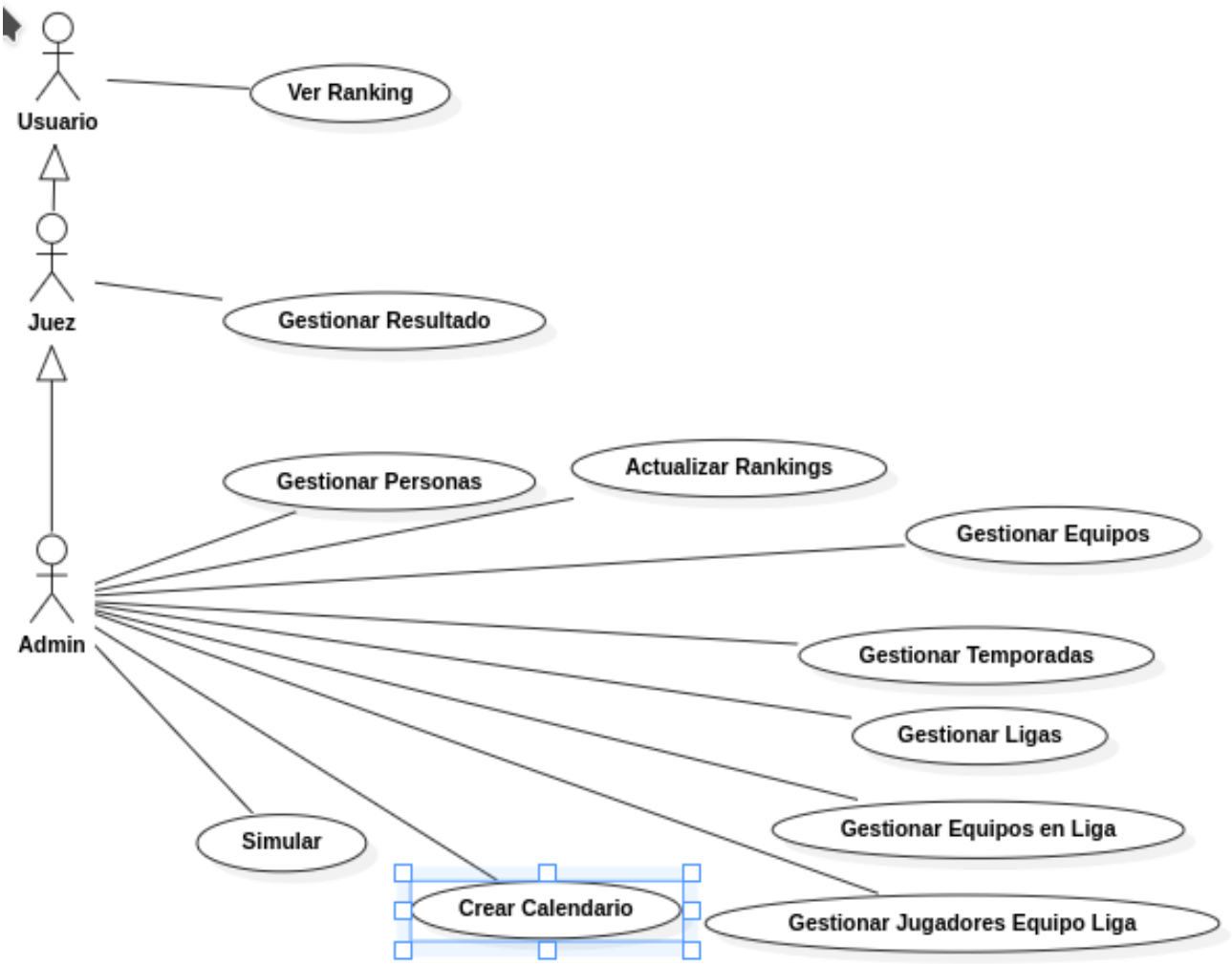

Figure 1.1: Caso de Uso de los Juegos

1.4 Diseño del Juego

Se proporciona el diagrama de clase que puede verse en la figura 1.2 que debe implementarse en los siguientes apartados.

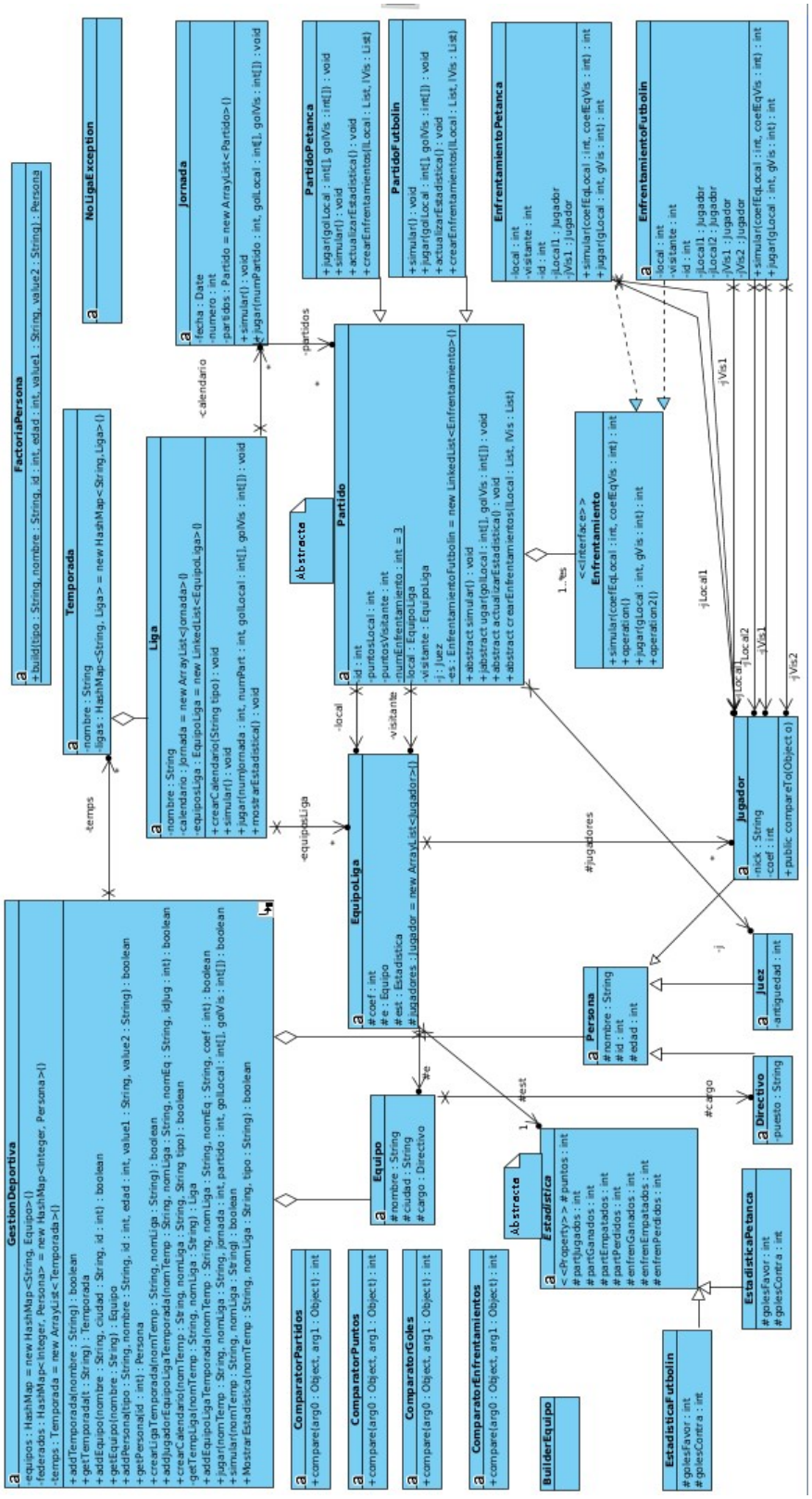

Figure 1.2: Diagrama de clase

1.5 Implementación

En este apartado se proporciona unos breves detalles sobre la implementación.

1.5.1 Especificaciones comunes:

El objetivo será la implementación del diagrama de clase anterior. La evaluación consistirá:

- Implementar cada una de las clases (según las indicaciones que se establecen en las siguientes secciones). Para cada una de las clases de deberá implementar los constructores, getters, setters, equals y toString. Se puede utilizar la generación automática de código. Se debe prestar mucha atención al nombre de los atributos pues los getters y setters cambian según su nombre.
- La implementación de la bateria de test sobre los métodos indicados de la clase Jornada (se entregará antes de la entrega final).
- Los comentarios Javadoc de todas las clases.
- El funcionamiento se comprobará en base a la ejecución de una bateria jUnit sobre la clase LNFutbolin, que proporcionará el profesorado de la asignatura. No se debe implementar ningún main. De este modo cuando se realice la tercera entrega, el sistema será completamente independiente de la vista que se use (consola, swing, html,...). No puede existir ningún syso en ninguna clase en la entrega.
- Se deberá implementar al menos un clase Exception en el proyecto denominada NoLigaException que se generará cuando se quiera recuperar una liga no existente para una temporada dada.
- Se deberá implementar una clase FactoriaPersona para la creación de las distintas personas. El último parámetro dado es de tipo String y es el coeficiente (hay que convertirlo) de cuando es un jugador.
- Se deberá usar el patrón BuilderEquipo para la creación de los Equipos en lugar de usar el constructor parametrizado. Esta clase será proporcionada por el profesorado.
- La entrega de la práctica será individual. Al realizarse la gestión de distintos campeonatos, al final del documento se establece qué campeonato debe implementar cada estudiante debiendo implementar las clases derivadas del campeonato que le ha tocado.
- Se puede definir métodos privados no recogidos en este diagrama de clase, para no alterar el comportamiento. Se debe consultar antes de añadir cualquier atributo a una clase o algún otro método público.

1.5.2 Jerarquía de Clase Persona

La jerarquía de clase Persona es una clase básica para modelar el concepto de una persona. El atributo id se usará para la identificación únivoca de cada persona (no puede haber dos personas con el mismo id). Esta formado por:

- nombre
- id
- edad

Se encuentra formada por las siguientes clases derivadas:

- Directivo que almacenará información del cargo
- Juez que almacenará la antigüedad como juez
- Jugador que almacenará el nick (como se le conoce) y un coeficiente (puntuación del jugador)

1.5.3 Clase Equipo

La clase equipo es una clase básica para modelar el concepto de un equipo en el campeonato. Esta formado por:

- Nombre
- Ciudad
- Directivo donde solo hay uno.

1.5.4 Jerarquía de Clase Abstracta Estadística

La jerarquía de clase Estadistica (abstracta) es una jerarquía básica para modelar todas las estadística de un equipo que participa en un determinada campeonato. Esta formado por los siguientes atributos: puntos, partJugados, partGanados, partEmpatados, partPerdidos, enfrenGanados, enfrenEmpatados, enfrenPerdidos. La clase abstracta se implementa en dos clases derivadas, que son:

- EstadisticaFutbolin: que tiene como atributos derivados golesFavor y golesContra. Los puntos se calculan: 3 puntos por partidos ganados, 1 punto por partido perdido y 0 por perdidos.
- EstadisticaPetanca: No tiene atributos derivadas. Los puntos se calculan: 5 puntos por partidos ganados, 2 punto por partido perdido y -3 por perdidos.

1.5.5 EquipoLiga

La clase EquipoLiga es una clase básica que se utiliza para representar cada una de las participaciones que ha tenido un determinado equipo en alguna liga de los campeonatos. Está formado:

- Coef que es un valor entero para representar el coeficiente del equipo.
- e que es el equipo asociado a este EquipoLiga (se debe implementar su getter y setter).
- est que representa las estadística para el equipoLiga (se debe implementar su getter y setter). Dependiendo del campeonato tendrá instanciado una EstadisticaFutbolin o una EstadisticaPetanca.
- jugadores que es un ArrayList con los jugadores que tiene ese EquipoLiga. Se debe implementar su: getter, setter, addJugador y getJugador (int id).

1.5.6 Interface Enfrentamiento

El interface Enfrentamiento es un interface que se utiliza para modelar cada uno de los enfrentamientos que hay en un partido. Tienen un comportamiento formado por dos métodos: simular y jugar. De este modo si se quiere añadir cualquier otro campeonato, solo se deberá implementar estos métodos que lo hace diferente. Está formado dos clases implementadoras.

1.5.6.1 Clase Implementadora EnfrentamientoFutbolin

Esta clase implementa el interface enfrentamiento. Está formada:

- id que es un identificador de los enfrentamientos.
- local que es la puntuación que ha obtenido el equipo local en este enfrentamiento
- visitante que es la puntuación que ha obtenido el equipo local en este enfrentamiento
- jLocal1 que es el jugador 1 del equipo local
- jLocal2 que es el jugador 2 del equipo local
- jVis1 que es el jugador 1 del equipo visitante
- jVis2 que es el jugador 2 del equipo visitante

El comportamiento de los dos métodos implementados es el siguiente:

- simular() que recibe el coeficiente del equipo local y del equipo visitante. Devuelve un valor entero donde se ha usado el coeficiente del equipo local junto con el coeficiente del jugador 1 y 2 local y se le ha restado el coeficiente del equipo visitante junto con el coeficiente del jugador 1 y 2 visitante. De este modo la simulación dependerá de los coeficientes de los jugadores y del equipo. Si devuelve un valor positivo el equipo local junto con sus jugadores tiene mejor coeficiente que el equipo visitante, si es negativo sería al revés y si vale 0 están empatados en coeficientes.
- jugar() que recibe el número de goles del equipo local y del equipo visitante y al igual que antes se devuelve la resta de estos valores. Esta operación se utilizará cuando el juez anote el resultado del enfrentamiento.

1.5.6.2 Clase Implementadora EnfrentamientoPetanca

Esta clase implementa el interface enfrentamiento. Está formada:

- id que es un identificador de los enfrentamientos.
- local que es la puntuación que ha obtenido el equipo local en este enfrentamiento
- visitante que es la puntuación que ha obtenido el equipo local en este enfrentamiento
- jLocal1 que es el jugador 1 del equipo local
- jVis1 que es el jugador 1 del equipo visitante

El comportamiento de los dos métodos implementados es el siguiente:

- simular() que recibe el coeficiente del equipo local y del equipo visitante. Devuelve un valor entero donde se ha usado el coeficiente del equipo local junto con el coeficiente del jugador y se le ha restado el coeficiente del equipo visitante junto con el coeficiente del jugador. De este modo la simulación dependerá de los coeficientes de los jugadores y del equipo. Si devuelve un valor positivo el equipo local junto con sus jugadores tiene mejor coeficiente que el equipo visitante, si es negativo sería al revés y si vale 0 están empatados en coeficientes.
- jugar() que recibe el número de puntos del equipo local y del equipo visitante y al igual que antes se devuelve la resta de estos valores. Esta operación se utilizará cuando el juez anote el resultado del enfrentamiento.

1.5.7 MainEnfrentamiento

Se recomienda, antes de continuar con la práctica, la implementación de un pequeño main para comprobar el correcto funcionamiento de Enfrentamiento. Se proporciona un main de ejemplo en el bitbucket público de la asignatura.

1.5.8 Jerarquía de Clase Partido

La clase Partido es una clase básica que se utiliza para representar cada uno de los partidos de una jornada. Dependiendo del campeonato seleccionado, se deberá instanciar como PartidoFutbolin o PartidoPetanca. Está formado:

- id que es un identificador de partido.
- local que es el EquipoLiga local del partido
- visitante que es el EquipoLiga visitante del partido
- puntosLocal que almacena el número de enfrentamientos que ha ganado el equipo local
- puntosVisitante que almacena el número de enfrentamientos que ha ganado el equipo visitante
- j que es el juez que arbitrará todos los enfrentamientos de este partido
- es que es una lista de los distintos enfrentamientos que tendrá el partido. Se debe implementar el método que permita añadir un enfrentamiento y obtener un enfrentamiento dado su id.
- static final int numEnfrentamiento que tendrá un valor fijo de 3 para indicar el número de enfrentamientos.

1.5.8.1 Clase Derivada PartidoFutbolin

Esta clase no tiene atributo y el comportamiento de los siguientes métodos es el siguiente:

- simular() que se encarga de realizar la simulación de todos los enfrentamientos del partido, actualizando los puntos de cada equipo. Además debe actualizar las estadísticas de los equipos con respecto al enfrentamiento: goles, enfrentamientos.
- jugar() que recibe dos arrays de entero con los goles de cada enfrentamiento. Tras cada enfrentamiento debe actualizar los puntos de cada equipo así como las estadísticas de los equipos con respecto al enfrentamiento: goles, enfrentamientos.
- actualizarEstadistica que será llamado tras la finalización de todos los enfrentamientos, es decir, cuando acaba el partido. Se encargará de actualizar las estadísticas de los equipos referidas: partidos y puntos. Este método es llamada desde Jornada.
- crearEnfrentamientos que se encargará de crear los distintos enfrentamientos del partido. En base al atributo del número de enfrentamientos que debe realizar se crearán tanto enfrentamientos como se indique. Recibe dos listas: una con los jugadores locales y otra con los jugadores visitantes. El sorteo de enfrentamientos se realizará desordenando estas listas y asignando los jugadores por posición, es decir, posición 0 con posición 0, posición 1 con posición 1 y así sucesivamente.

1.5.8.2 Clase Derivada PartidoPetanca

Esta clase no tiene atributo y el comportamiento de los siguientes métodos es el siguiente:

- simular() que se encarga de realizar la simulación de todos los enfrentamientos del partido, actualizando los puntos de cada equipo. Además debe actualizar las estadísticas de los equipos con respecto al enfrentamiento: goles, enfrentamientos.
- jugar() que recibe dos arrays de entero con los goles de cada enfrentamiento. Tras cada enfrentamiento debe actualizar los puntos de cada equipo así como las estadísticas de los equipos con respecto al enfrentamiento: enfrentamientos.
- actualizarEstadistica que será llamado tras la finalización de todos los enfrentamientos, es decir, cuando acaba el partido. Se encargará de actualizar las estadísticas de los equipos referidas: partidos y puntos. Este método es llamada desde Jornada.
- crearEnfrentamientos que se encargará de crear los distintos enfrentamientos del partido. En base al atributo del número de enfrentamientos que debe realizar se crearán tanto enfrentamientos como se indique. Recibe dos listas: una con los jugadores locales y otra con los jugadores visitantes. El sorteo de enfrentamientos se realizará ordenando en primer lugar las dos listas por el coeficiente del jugador y asignando los jugadores por el orden establecido: el mejor de un equipo por el el mejor del otro, y así sucesivamente.

1.5.9 MainPartido

Se recomienda, antes de continuar con la práctica, la implementación de un pequeño main para comprobar el correcto funcionamiento de Partido. Se proporciona un main de ejemplo en el bitbucket público de la asignatura.

1.5.10 Jornada

La clase Jornada es una clase que se utiliza para representar cada una de las jornadas de una liga. Está formado:

- numero que es un identificador de jornada.
- fecha que representa cuando se juega la jornada
- partidos que es una lista de los distintos partidos que tendrá la jornada. Se debe implementar el método que permita añadir un partido y obtener un partido dado su id.

Esta clase tiene los siguientes métodos importantes para su funcionamiento:

- simular() que se encarga de realizar la simulación de todos los partidos de la jornada. Tras cada partido, se debe actualizar las estadísticas.
- jugar() que recibe dos arrays de entero con los goles para el partido indicado por parametro. Tras el partido, se debe actualizar las estadísticas.

1.5.11 MainJornada

Se recomienda, antes de continuar con la práctica, la implementación de un pequeño main para comprobar el correcto funcionamiento de Jornada. En este caso se recomienda al estudiante que haga su propia implementación.

1.5.12 Test de batería Jornada

Para probar el funcionamiento correcto de la implementación de Jornada se deberá realizar y entregar antes de la fecha de la segunda entrega final una batería de test de Jornada.

1.5.13 Liga

La clase Liga es una clase que se utiliza para modelar cada una de las ligas de una temporada. Está formado:

- nombre que es un identificador de la liga.
- calendario que es una lista de las distintas jornadas que tendrá la liga. Se debe implementar el método que permita añadir una jornada y obtener una jornada dado el número de la jornada.
- equiposLiga que es una lista de los distintos EquipoLiga que tendrá la liga. Se debe implementar el método que permita añadir un EquipoLiga y obtener un EquipoLiga dado el nombre de un equipo.

Esta clase tiene los siguientes métodos importantes para su funcionamiento:

- simular() que se encarga de realizar la simulación de todos las jornadas de la liga.
- jugar() que recibe dos arrays de entero con los goles para la jornada indicada para el número de partido indicado.
- mostrarEstadistica que se encarga de mostrar las estadísticas.
- crearCalendario(String tipo) que se encarga de realizar el reparto de jornadas y partidos para los equipos existente en la liga. Se proporcionará el esqueleto principal de esta función a falta de algunas sentencias. En base al tipo("Futbolin" o "Petanca", se deberá crear un PartidoFutbolin o PartidoPetanca. IMPORTANTE: En este paso, una vez que se crea el calendario, es cuando fijaremos el tipo de estadística también.

El comportamiento crearCalendario es el siguiente (algoritmo 1) que se ha obtenido adaptando el código mostrado aquí[1].

[1] obtenido de: //https://stackoverflow.com/questions/26471421/round-robin-algorithm-implementation-java

Algorithm 1 Operación de CrearCalendario

```
public void crearCalendario() {
        int numTeams = equiposLiga.size();
        int numDays = (numTeams - 1); // Dias que tendrá el torneo
        int halfSize = numTeams / 2; //calculo e la mitad
        //Lista para copiar la lista de jugadores
        List<EquipoLiga> teams = new ArrayList<EquipoLiga>();
        teams.addAll(equiposLiga); // Añado todos los equipos al temporal
        teams.remove(teams.get(0)); //Elimino el primer elemento
        int teamsSize = teams.size();
        for (int day = 0; day < numDays; day++) {
            //Primer partido de la jornada
            System.out.println("Jornada: " + day);
            int teamIdx = day % teamsSize;
            System.out.println(teams.get(teamIdx) + " vs " + equiposLiga.get(0));
            //TODO: crear jornada
            //TODO: crear primer partidos
            //TODO: fijar la estadistica en base al tipo
           //TODO: Asignar Juez
            //TODO: crear enfrentamiento para cada partido
            //El resto de partidos de la jornada
            for (int idx = 1; idx < halfSize; idx++) {
              int firstTeam = (day + idx) % teamsSize;
              int secondTeam = (day + teamsSize - idx) % teamsSize;
              System.out.println(teams.get(firstTeam) + " vs " + teams.get(secondTeam));
               //TODO: crear cada partido
            //TODO: fijar la estadistica en base al tipo
             //TODO: Asignar Juez
              //TODO: crear enfrentamiento para cada partido
            }
            calendario.add(j);
        }
}
```

1.5.14 MainLiga

Se recomienda, antes de continuar con la práctica, la implementación de un pequeño main para comprobar el correcto funcionamiento de Liga. En este caso se recomienda al estudiante que haga su propia implementación.

1.5.15 Temporada

La clase Temporada es una clase que se utiliza para modelar cada una de las temporadas. Está formado:

- nombre que es un identificador de la temporada.
- ligas que es una lista de las distintas ligas que tendrá la temporada. Se debe implementar el método que permita añadir una liga y obtener una liga dada su nombre.

1.5.16 Clase LNFutbolin

Se encuentra formado por las Personas, Equipos, temporadas y el tipo de campeonato que se quiere jugar ("Futbolin" o "Petanca"). Se debe implementar las siguientes operaciones:

- Constructor
- boolean addTemporada(String nombre)
- Temporada getTemporada(String t)
- boolean addEquipo(String nombre, String ciudad, int id)
- Equipo getEquipo(String nombre)
- boolean addPersona
- Persona getPersona(int id)
- boolean crearLigaTemporada(String nomTemp, String nomLiga)
- boolean addEquipoLigaTemporada(String nomTemp, String nomLiga, String nomEq, int coef): Se tiene en cuenta el tipo de campeonato para crear una estadística u otra.
- boolean addJugadorEquipoLigaTemporada(String nomTemp, String nomLiga, String nomEq, int idJug)
- boolean crearCalendario(String nomTemp, String nomLiga,String tipo)
- boolean jugar(String nomTemp, String nomLiga, int jornada, int partido, int []golLocal, int []golVis)
- boolean simular(String nomTemp, String nomLiga)
- boolean MostrarEstadistica(String nomTemp, String nomLiga, String tipo) donde tipo puede "Puntos", "Partidos", "Enfrentamientos" y "Goles. En base al tipo recibido, se deberá ordenar los equipos en base a la clasificación solicitada.

1.6 Asignación de Entrega Final

En este apartado se asigna la implementación que debe realizar cada estudiante.

Table 1.1: Asignación de Trabajos

Futbolín	**Petanca**
JUAN LUIS ARENAS SÁNCHEZ	MARTA XIAO BAHAMONDE OSUNA
JULIÁN BLANCO GONZÁLEZ	JAIME LUIS CALVO ARDILA
ANTONIO CALVO PICÓN	JOSÉ LUIS OBIANG ELA NANGUAN
GUILLERMO FERNÁNDEZ RUBIO	JAVIER GARCÍA SÁNCHEZ
DAVID GIL PAREJO	RUBÉN GONZÁLEZ GARCÍA
DIEGO GONZÁLEZ RODRÍGUEZ	LUIS IGNACIO HERNÁNDEZ MORANO
EVARISTO MANCERA LLANO	JUAN PÉREZ VILLEGAS
GUILLERMO PINILLA CARRASCAL	JOSÉ LUIS PLATA GALLARDO
JAVIER PRADA NAHARROS	JUAN CARLOS REDONDO PRIOR
JAVIER REY SÁNCHEZ	ANTONIO JAVIER RINO CIDONCHA
AIDA RODRÍGUEZ UGARTE	DANIEL ROMERO LOZANO
IGNACIO SÁNCHEZ AGUILERA	IVÁN SAMPÉ PRIETO
PABLO SETRAKIAN BEARZOTTI	IVÁN TREJO LOZANO
DAVID URBANO RANCHAL	ENRIQUE VIDAL VICENTE