Martí Esquius Arnau, u213923, marti.esquius01@estudiant.upf.edu
Marc Guiu Armengol, u215107, marc.guiu01@estudiant.upf.edu

Lab2 Documentation

En esta práctica hemos implementado 2 clases nuevas y ampliado las 2 de la práctica pasada (Player y Team), al final de la práctica añadimos la explicación de estas dos clases, puesto que ya fueron implementadas en la sesión anterior pero han sido modificadas parcialmente.

En primer lugar hemos implementado una primera clase llamada **Match** que sirve para disputar partidos entre 2 equipos, con los atributos siguientes:

- **homeTeam:** Team -> Uno de los equipos que se disputan el partido, el local.
- **awayTeam:** Team -> El otro equipo, visitante.
- **homeGoals:** int -> Goles hechos por el equipo local.
- **awayGoals:** int -> Goles hechos por el equipo visitante
- **homeScorers:** list of Player -> Linked List que almacena los goleadores (jugadores que marcan goles) del equipo local.
- **awayScorers:** list of Player -> Linked List que almacena los goleadores (jugadores que marcan goles) del equipo visitante.

Y los métodos:

- **Match**(h:Team, a:Team) -> Método constructor de la clase, recibe 2 equipos a los que trata como local y visitante. Asigna a homeGoals y awayGoals un valor de 0, e inicializa las dos LinkedList que almacenarán las listas de los jugadores goleadores respectivamente.
- **getHomeTeam**() -> Método del tipo “getter” que retorna el equipo local de un partido en concreto.
- **getAwayTeam**() -> Método del tipo “getter” que retorna el equipo visitante de un partido en concreto.
- **getHomeGoals**() -> Método del tipo “getter” que retorna los goles del equipo local dado un partido.
- **getAwayGoals**() -> Método del tipo “getter” que retorna los goles del equipo visitante dado un partido.

- **getHomeScorers**() -> Método del tipo “getter” que retorna la Linked List que almacena los goleadores del equipo local.
- **getAwayScorers**() -> Método del tipo “getter” que retorna la Linked List que almacena los goleadores del equipo visitante.
- **simulateMatch**() -> Este método genera un partido con un numero aleatorio de goles entre 0 y 6 para cada equipo. Una vez obtenidos los goles copia de forma aleatoria tantos jugadores como sea necesario ‘x’ desde el equipo que ha marcado ‘x’ goles (clase que contiene una Linked List de jugadores) hasta la linked list de goleadores dentro del Match.

  Finalmente hace un updateStats de cada equipo pasando el partido como atributo.
- **printMatch**() -> Este método imprime todo lo relacionado con un partido, los equipos que están jugando, la puntuación final y los goleadores de cada equipo.

Y la clase **League,** que agrupa a los equipos y hace que disputen partidos entre ellos. Con los atributos:

- **name**: String -> El nombre de la liga.
- **country**: Country -> El país en el que se juega.
- **gender**: Gender -> El género de los jugadores (puede ser masculino, femenino o mixto).
- **teams**: list of Team -> Lista con los equipos que hay en la liga
- **matches**: list of Match -> Lista con los partidos que se tienen que jugar en la liga.

Y los métodos:

- **League**(n:String, c:Country, g:int) -> Método constructor. Recibe un string que lo asigna al nombre, un Country que lo asigna al country y un int que lo asigna al género. Inicializa ambas listas (teams y matches) para que después con otros métodos se puedan modificar.
- **getName**() -> Método del tipo “getter” que retorna el nombre de la liga en cuestión.
- **getCountry**() -> Método del tipo “getter” que retorna el país de la liga.
- **getGender**() -> Método del tipo “getter” que retorna el género de los jugadores que pueden jugar en la liga.
- **getTeams**() -> Método del tipo “getter” que retorna la lista de equipos de la liga.
- **getMatches**() -> Método del tipo “getter” que retorna la lista de partidos que se tienen que disputar-

- **addTeam**(t:Team) -> Método que recibe un Team y lo añade a la lista teams de equipos si cumple los requisitos necesarios para estar en la liga (país y género).
- **generateMatches**() -> Método que genera la lista de matches, emparejando a todos los equipos con todos 2 veces (ida y vuelta) donde en el primero uno es local y el otro visitante y en el segundo se intercambian los papeles.
- **simulateMatches**() -> Método que simula todos los partidos de la lista, uno por uno usando las funciones del método anterior para ello. Asigna un ganador y un perdedor a cada partido.
- **printMatches**() -> Método que imprime los partidos de la lista con sus respectivos resultados.

Estas son clases ya explicadas en la sesión 1, pero han sido utilizadas y modificadas(*) también para esta segunda sesion, clase **Player**, con los siguientes atributos:

- **female**: bool -> Nos dice el género del jugador (TRUE si es chica y FALSE si es chico)
- **name**: String -> Nos dice el nombre del jugador en un String.
- **age**: int -> Nos dice la edad del jugador.
- **nationality**: Country -> Nos dice la nacionalidad del jugador, en la clase, que ya nos venía programada, Country.
- **noMatches**: int -> Nos dice el número de partidos que lleva el jugador en un int.
- **noTackles**: int -> Nos dice el número de entradas que lleva el jugador en un int.
- **noPasses**: int -> Nos dice el número de pases que lleva el jugador en un int.
- **noShots**: int -> Nos dice el número de disparos a portería que lleva el jugador en un int.
- **noAssists**: int -> Nos dice el número de asistencias que lleva el jugador en un int.
- **noGoals**: int -> Nos dice el número de goles que lleva el jugador en un int.

A partir de estos atributos, hemos creado los siguientes métodos:

- **Player**(g:bool, n:String, a:int, nat:Country) -> Este es el método constructor, inicializa todas los atributos de la clase, asignando las variables recibidas a sus correspondientes atributos (female, name, age y nationality respectivamente), y los demás los inicializa a 0 (ya que se supone que todavía no ha jugado ningún partido).

- **isFemale**(): bool -> Este método, nos dice el género del jugador, diciéndonos si es o no chica (TRUE o FALSE).
- **getName**(): String -> Nos devuelve un String con el nombre del jugador.
- **getAge**(): int -> Nos devuelve un int con la edad del jugador.
- **getNationality**(): Country -> Nos devuelve una clase Country con la nacionalidad del jugador.
- **update**(t:int, p int, s int, a int, g int) -> Actualiza las estadísticas del jugador después de un partido. Suma uno a noMatches, y suma t, p, s, a y g a noTackles, noPasses, noShots, noAssists y noGoals respectivamente.
- **printStats**() -> Imprime las estadísticas del jugador.
- **(NUEVO!)updateStats**(m:Match) -> Recibe un partido y aumenta en uno los partidos jugados por el jugador (es un método que llamaremos al final de simular un partido). Además, recorre las listas de goleadores de ambos equipos y si el jugador actual está presente en alguna de las listas aumenta su número de goles una vez por cada gol.

La otra clase que ha sido modificada es **Team**, con los atributos:

- **name**: String -> Nos dice el nombre del equipo en un String
    - **country**: Country -> Nos dice el país del equipo en una clase Country.
- **gender**: Gender-> Nos dice el género de los jugadores (MALE, FEMALE o MIXT), en una enumeración que hemos definido previamente en el archivo Gender.java.
- **players**: list of Player -> Es una lista de los jugadores del equipo dónde cada uno de ellos es una clase Player.
- **noMatches**: int -> Nos dice el número de partidos que lleva el equipo en un int.
- **noWins**: int -> Nos dice el número de victorias que lleva el equipo en un int.
- **noTies**: int -> Nos dice el número de empates que lleva el equipo en un int.
- **noLosses**: int -> Nos dice el número de derrotas que lleva el equipo en un int.
- **goalsScored**: int -> Nos dice el número de goles a favor que lleva el equipo en un int.
- **goalsAgainst:** int -> Nos dice el número de goles en contra que que lleva el equipo en un int.

Con estos atributos hemos creado los siguientes métodos:

- **Team**(n:String, c:Country, g:Gender) -> Este es el método constructor, que inicializa todos los atributos del equipo, donde n es un String que se asocia al nombre (name), c es una clase Country que se asocia al país (Country) y g es una de las posibilidades de la enumeración Género que se asigna al género del equipo

(gender). Los otros atributos se inicializan en 0 ya que se supone que el equipo no ha jugado ningún partido todavía. También inicializa la lista de jugadores.

- **getName**(): String -> Nos devuelve un String con el nombre del equipo.
- **getCountry**(): Country -> Nos devuelve una clase Country con el país del equipo.
- **getGender**(): Gender -> Nos devuelve el género del equipo (una de las tres posibilidades de la enumeración).
- **getPlayer**(n:int): Player-> Nos devuelve el jugador con el indice n de la Linked List.
- **getWins**(): int -> Nos devuelve el número de victorias.
- **getTies**(): int -> Nos devuelve el número de empates.
- **getLosses**(): int -> Nos devuelve el número de derrotas.
- **addPlayer**(Player p): boolean -> Si se cumplen las condiciones (mismo género del equipo y del jugador, o por otro lado que el equipo sea mixto), añade al jugador en el equipo y devuelve TRUE para verificar que se ha podido hacer, en caso contrario simplemente devuelve FALSE.
- **removePlayer**(Player p) -> Elimina a un jugador de la lista de jugadores.
- **playMatch**(fgoals:int, agoals:int) -> Actualiza los atributos del equipo después de un partido, sumando uno a noMatches, añadiendo los goles a favor (fgoals) y en contra (agoals) y con un if, mira cual de los dos números es más grande para decidir si tiene que sumar 1 a las victorias (noWins), a las derrotas (noLosses) o a los empates (noTies).
- **printStats**() -> Imprime las estadísticas del equipo.
- **(NUEVO!)updateStats**(m:Match) -> Hace un updateStats de cada uno de los jugadores del equipo y llama a la función playMatch después de saber si es el equipo visitante o local para que allí se actualizen las estadísticas del equipo.