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

Lab3 Documentation

En esta práctica, hemos añadido las siguientes clases: NationalTeam (subclase de team), Goalkeeper y Outfielder (subclases de Player), League, Group Play y Cup (subclases de Competition) y cup Match (subclase de Match). Todas ellas guardan una relación de herencia con sus superclases, por lo que heredan sus atributos. En algunos casos, se añaden nuevos atributos en la subclase y en otros simplemente se añaden o reescriben métodos para cambiar alguna funcionalidad. A continuación hablaremos detalladamente de cada una de estas subclases y en el Anexo se pueden encontrar las clases de prácticas pasadas con las actualizaciones pertinentes.

En primer lugar, NationalTeam no tiene atributos propios y tiene los siguientes métodos:

- **NationalTeam**(n:string, c:country, g:int) -> Método constructor, el nombre, el país y el género de los jugadores, e inicializa los atributos de Team con ello. Se usará country para verificar la nacionalidad de los jugadores.
- **addPlayer**(p:Player) -> Este método recibe un jugador y no añade al equipo, verificando que la nacionalidad sea la misma, ya que se trata de un equipo nacional.

En segundo lugar, tenemos la clase Goalkeeper, que es una subclase de jugador, que tiene los siguientes atributos:

- **noSaves**: int -> Este atributo sirve para guardar el número de paradas que ha hecho el portero.
- **noGoalsLet**: int -> En este se guarda el número de goles que este ha recibido.

Y los siguientes métodos:

- **Goalkeeper**(g:int ,n:string ,a:int ,nat:Country) -> Este es el método constructor, recibe el género, nombre, edad y nacionalidad del portero e inicializa a 0 los otros atributos.
- **updateStats**(m:Match) -> Este método sirve para actualizar las estadísticas del portero después de un partido, aumenta el número de partidos y calcula el de paradas.

Y Outfielder que es otra subclase de jugador, con los siguientes atributos:

- **noTackles**: int -> Este atributo sirve para almacenar el número de entradas del jugador.
- **noPasses**: int -> Este atributo sirve para almacenar el número de pases del jugador.
- **noShots**: int -> Este atributo sirve para almacenar el número de disparos a portería del jugador.
- **noAssists**: int -> Este atributo sirve para almacenar el número de asistencias (pases de gol) del jugador.
- **noGoals**: int -> Este atributo sirve para almacenar el número de goles del jugador.

Y los métodos:

- **Outfielder**(g:int, n:string, a:int, nat:Country) -> Este es el método constructor, recibe el género, nombre, edad y nacionalidad del jugador e inicializa a 0 los otros atributos.

- **updateStats**(m:Match) -> Este método sirve para actualizar las estadísticas del jugador después de un partido, aumenta el número de partidos y el número de goles para sumarlo

.

A continuación tenemos las subclases de Competition. La primera de ellas es League, sin atributos y con los métodos:

- **League**(n:string, c:Country, g:int) -> Este es el método constructor, recibe el nombre y país de la liga y el género de los jugadores que juegan en ella. También inicializa el resto de atributos de la liga, como la lista con los equipos.
- **generateMatches**() -> Este método sirve para generar los partidos de la liga, emparejando a todos los equipos con todos 2 veces (ida y vuelta).

La segunda es GroupPlay, que en este caso sí que tiene atributos:

- **noGroups**: int -> Este atributo almacena el número de grupos que hay en la competición, donde cada uno de ellos es una liga.
- **groups**: array of League -> En este atributo se almacenan cada una de las ligas previamente mencionadas.

Y métodos:

- **GroupPlay**(n:string, c:Country, g:int) -> Este es el método constructor, recibe el nombre y país de la competicion y el género de los jugadores que juegan en ella. También inicializa el resto de atributos.
- **generateMatches**() -> Este método genera los partidos, generando los grupos y empareja los equipos de cada uno de estos grupos como se hace en una liga.
- **simulateMatches**() -> Este método simula los partidos de la liga, dónde el resultado puede ser victoria, derrota o empate.

Y en tercer lugar tenemos la subclase Cup, con los atributos:

- **tr**: array of list of Team -> Almacena los equipos de cada ronda de la copa, y a medida que se va avanzando, deja de haber los equipos que han sido eliminados.
- **mr**: array of list of Match -> Almacena los partidos de cada ronda de la copa.

Y los métodos:

- **Cup**(n:string, c:Country, g:int) -> Este es el método constructor, recibe el nombre y país de la copa y el género de los jugadores que juegan en ella. También inicializa las listas para poderlas usar más tarde.
- **generateMatches**() -> Este método genera los partidos de una determinada ronda (emparejando los equipos que quedan 2 a 2).
- **simulateMatches**() -> Este método simula cada uno de los partidos, pasando al ganador a la siguiente ronda y descalificando al perdedor.

Finalmente, tenemos CupMatch, subclase de Match, que no tiene atributos y tiene los siguientes métodos:

- **cupMatch**(h:Team, a:Team) -> Es el método creador, recibe 2 equipos y asigna uno como local y otro como visitante, también inicializa los otros atributos.
- **simulateMatch**() -> Simula los partidos, en este caso el empate no es una opción, por lo que si se da, simula una prórroga, y en caso extremo, si es oportuno, unos penaltis para desempatar.

Anexo

Match:

Atributos:

- **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.

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 número 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.

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

Competition (en la práctica anterior era League):

Atributos:

- **(NUEVO) clubs**: bool -> Indica si es una competición de clubes o de selecciones.
- **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.

Métodos:

- **Competition**(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.

Player (se han eliminado algunos atributos que han pasado a ser específicos de su subclase Outfielder):

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.

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.
- **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.

Team:

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.

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 índice 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..
- **printStats**() -> Imprime las estadísticas del equipo.
- **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.