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

Lab1 Documentation

En esta práctica, hemos implementado dos clases nuevas, además de la que ya nos daban.

La primera es la 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.

Y una segunda clase **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).
- **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.

Posibles mejoras del código serían, por ejemplo, actualizar las estadísticas de cada jugador del equipo dentro del método playMatch, y más allá de mejoras, el código se podría continuar añadiendo nuevas clases relacionadas como ligas o selecciones, expandiendo así las funcionalidades del código.

Una vez implementadas las dos clases, hemos hecho una prueba para comprobar que todo funcionaba correctamente, creando una serie de jugadores y de equipos para probar todos los métodos creados a lo largo de todo el código. El resultado ha sido positivo y todos los métodos han tenido el comportamiento esperado.