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

Lab4 Documentation

En esta práctica, hemos ampliado las clases existentes y añadido nuevas clases, así como métodos atributos como pedía el enunciado. A continuación, se presenta una descripción detallada de las actualizaciones en las clases, en el anexo se puede encontrar las clases con todos sus atributos y métodos, que siguen igual que en el laboratorio anterior:

1. Clase Team:

La clase Team ahora incluye un nuevo atributo y métodos relacionados con estadísticas de equipo para distintas competiciones. El nuevo atributo es:

- **stats**: Dictionary of (Competition, TeamStats) -> Almacena las estadísticas del equipo para cada competición en la que participa.

Los nuevos métodos son:

- **update**(c:Competition, m:Match): Actualiza las estadísticas del equipo después de un partido.

2. Clase TeamStats:

La clase TeamStats está diseñada para contener las estadísticas específicas de un equipo en una competición dada. Sus atributos son:

- **team**: Team -> Referencia al equipo asociado.
- **noMatches**: int -> Número total de partidos jugados.
- **noWins**: int -> Número de victorias.
- **noTies**: int -> Número de empates.
- **noLosses**: int -> Número de derrotas.
- **goalsScored**: int -> Número total de goles marcados.
- **goalsAgainst**: int -> Número total de goles en contra.

Los métodos incluyen:

- **TeamStats**(t:Team): Constructor que inicializa las estadísticas para un equipo específico.
- **updateStats**(m:Match): Actualiza las estadísticas del equipo después de un partido.
- **printStats**(): Imprime las estadísticas del equipo.
- **compareTo**(o:Object): int: Compara las estadísticas de dos equipos.

3. Clase Player:

La clase Player ahora incluye un nuevo atributo y métodos relacionados con estadísticas de jugador para distintas competiciones. El nuevo atributo es:

- **stats**: Dictionary of (Competition, PlayerStats) -> Almacena las estadísticas del jugador para cada competición en la que participa.

Los nuevos métodos son:

- **update**(c:Competition, m:Match): Actualiza las estadísticas del jugador después de un partido.
- **getStats**(c:Competition): Obtiene las estadísticas del jugador para una competición específica.

4. Clase PlayerStats (Clase Abstracta):

La clase PlayerStats es una clase abstracta diseñada para contener las estadísticas específicas de un jugador en una competición dada. Sus atributos son:

- **player**: Player -> Referencia al jugador asociado.
- **noMatches**: int -> Número total de partidos jugados.

Los métodos incluyen:

- **PlayerStats**(p: Player): Constructor que inicializa las estadísticas para un jugador específico.
- **updateStats**(m:Match): Actualiza las estadísticas del jugador después de un partido.
- **abstract printStats**(): Método abstracto para imprimir las estadísticas del jugador.

5. Clase Goalkeeper y GoalkeeperStats:

La clase Goalkeeper es una subclase de Player y ahora incluye un nuevo método. La clase GoalkeeperStats es una subclase de PlayerStats diseñada específicamente para guardametas. Sus atributos son:

- **noSaves**: int -> Número de paradas realizadas por el portero.
- **noGoalsLet**: int -> Número de goles recibidos por el portero.

El nuevo método es:

- update(c:Competition, m:Match): Actualiza las estadísticas del portero después de un partido.

6. Clase Outfielder y OutfielderStats:

La clase Outfielder es otra subclase de Player y ahora incluye nuevos atributos. La clase OutfielderStats es una subclase de PlayerStats diseñada específicamente para jugadores de campo. Sus atributos son:

- **noTackles**: int -> Número de entradas realizadas por el jugador.
- **noPasses**: int -> Número de pases realizados por el jugador.
- **noShots**: int -> Número de disparos a portería realizados por el jugador.
- **noAssists**: int -> Número de asistencias realizadas por el jugador.
- **noGoals**: int -> Número de goles marcados por el jugador.

Los métodos incluyen:

- **update**(c:Competition, m:Match): Actualiza las estadísticas del jugador después de un partido.

Anexo

Clase NationalTeam:

Atributos: No tiene atributos propios.

Métodos:

- NationalTeam(n:String, c:Country, g:int): Método constructor que inicializa el nombre, país y género de los jugadores, e inicializa los atributos de Team con ello. Utiliza el atributo country para verificar la nacionalidad de los jugadores.
- addPlayer(p:Player): Recibe un jugador y lo añade al equipo, verificando que la nacionalidad sea la misma, ya que se trata de un equipo nacional.

Clase Goalkeeper:

Atributos:

- noSaves: int: Número de paradas realizadas por el portero.
- noGoalsLet: int: Número de goles recibidos por el portero.

Métodos:

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

Clase Outfielder:

Atributos:

- noTackles: int: Número de entradas realizadas por el jugador.
- noPasses: int: Número de pases realizados por el jugador.
- noShots: int: Número de disparos a portería realizados por el jugador.
- noAssists: int: Número de asistencias (pases de gol) realizadas por el jugador.
- noGoals: int: Número de goles marcados por el jugador.

Métodos:

- Outfielder(g:int, n:string, a:int, nat:Country): Método constructor que recibe el género, nombre, edad y nacionalidad del jugador e inicializa a 0 los otros atributos.
- updateStats(m:Match): Actualiza las estadísticas del jugador después de un partido, aumenta el número de partidos y el número de goles para sumarlo.

Clase League:

Atributos: No tiene atributos propios.

Métodos:

- League(n:string, c:Country, g:int): Método constructor que 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(): Genera los partidos de la liga, emparejando a todos los equipos con todos 2 veces (ida y vuelta).

Clase GroupPlay:

Atributos:

- noGroups: int: Número de grupos en la competición, donde cada uno de ellos es una liga.
- groups: array of League: Almacena cada una de las ligas previamente mencionadas.

Métodos:

- GroupPlay(n:string, c:Country, g:int): Método constructor que recibe el nombre y país de la competición y el género de los jugadores que juegan en ella. También inicializa el resto de atributos.
- generateMatches(): Genera los partidos, generando los grupos y emparejando los equipos de cada uno de estos grupos como se hace en una liga.
- simulateMatches(): Simula los partidos de la liga, donde el resultado puede ser victoria, derrota o empate.

Clase Cup:

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.

Métodos:

- Cup(n:string, c:Country, g:int): Método constructor que recibe el nombre y país de la copa y el género de los jugadores que juegan en ella. También inicializa las listas para poder usarlas más tarde.

- generateMatches(): Genera los partidos de una determinada ronda (emparejando los equipos que quedan 2 a 2).
- simulateMatches(): Simula cada uno de los partidos, pasando al ganador a la siguiente ronda y descalificando al perdedor.

Clase CupMatch:

Atributos: No tiene atributos propios.

Métodos:

- cupMatch(h:Team, a:Team): Método creador que 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 penales para desempatar.

Player:

Atributos:

- female: bool: Indica el género del jugador (TRUE si es mujer y FALSE si es hombre).
- name: String: Nombre del jugador en formato String.
- age: int: Edad del jugador en formato entero.
- nationality: Country: Nacionalidad del jugador, representada por la clase Country.
- stats: Dictionary of (Competition, PlayerStats) -> Almacena las estadísticas del jugador para cada competición en la que participa.

Métodos:

- Player(g:bool, n:String, a:int, nat:Country): Método constructor que inicializa todos 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: Retorna TRUE si el jugador es mujer, FALSE si es hombre.
- getName(): String: Retorna el nombre del jugador en formato String.
- getAge(): int: Retorna la edad del jugador en formato entero.
- getNationality(): Country: Retorna la nacionalidad del jugador, representada por la clase Country.
- update(c:Competition, m:Match): Actualiza las estadísticas del jugador después de un partido.

Team:

Atributos:

- name: String: Nombre del equipo en formato String.
- country: Country: País del equipo representado por la clase Country.

- gender: Gender: Género de los jugadores (MALE, FEMALE o MIXT), en una enumeración previamente definida.
- players: list of Player: Lista de los jugadores del equipo, donde cada uno de ellos es una instancia de la clase Player.

Métodos:

- Team(n:String, c:Country, g:Gender): Método constructor que inicializa todos los atributos del equipo. n se asocia al nombre (name), c se asocia al país (country), y g se asocia al género del equipo (gender).
- getName(): String: Retorna el nombre del equipo en formato String.
- getCountry(): Country: Retorna el país del equipo, representado por la clase Country.
- getGender(): Gender: Retorna el género del equipo (una de las tres posibilidades de la enumeración).
- getPlayer(n:int): Player: Retorna el jugador con el índice n de la Linked List.
- getWins(): int: Retorna el número de victorias.
- getTies(): int: Retorna el número de empates.
- getLosses(): int: Retorna el número de derrotas.
- addPlayer(Player p): boolean: Añade al jugador en el equipo si cumple los requisitos necesarios para estar en la liga (mismo género del equipo y del jugador, o por otro lado que el equipo sea mixto). Retorna TRUE si se ha podido hacer, en caso contrario, retorna FALSE.
- removePlayer(Player p): Elimina a un jugador de la lista de jugadores.
- update(c:Competition, m:Match): Actualiza las estadísticas del equipo después de un partido.

Goalkeeper:

Atributos: No tiene

Métodos:

- update(c:Competition, m:Match): Actualiza las estadísticas del portero después de un partido.

Outfielder:

Atributos: No tiene

Métodos:

- update(c:Competition, m:Match): Actualiza las estadísticas del jugador después de un partido.