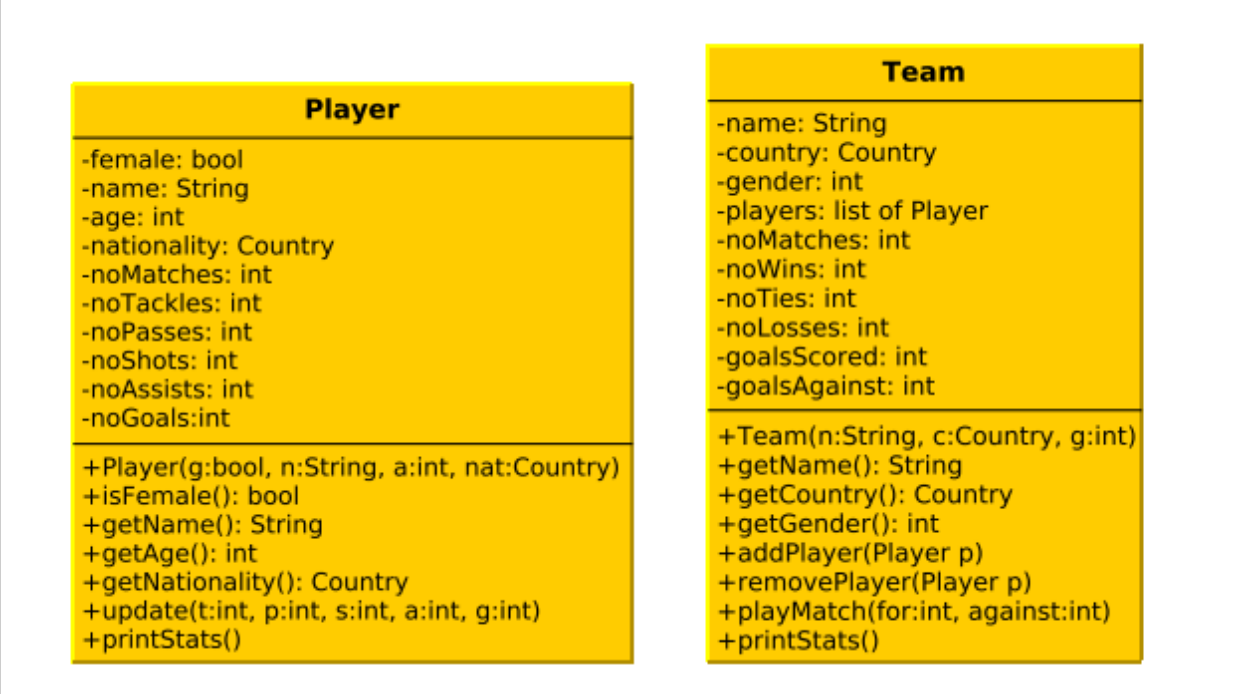
Lab1. Programació Orientada a Objectes

Jhonatan Barcos Gambaro - P201 G09

Índice

1. Introducción

Este proyecto se basa en el seminario 1 realizado previamente en el cual diseñamos dos clases: **Team** (Equipo de fútbol) y **Player** (Jugador de fútbol).



A partir de dicho diseño, hemos realizado un programa en el cual se permita la creación, gestión y seguimiento de equipos, jugadores y partidos de fútbol.
Vayamos clase por clase detallando sus componentes:

- **Team:**

Atributos: *name, country, genderClass, players, noMatches, noWins, noTies, noLosses, noGoalsScored, y noGoalsReceived*.
Métodos: La clase Team consta de un constructor *Team(...)* que inicializa un equipo con su nombre, país y género. A continuación una parte del código implementado para esta clase:

```java
public class Team{
    //Atributos
    private String name;
    private Country country;
    private Gender genderClass;
    private LinkedList<Player> players = new LinkedList<Player>();
    private int noMatches;
    private int noWins;
    private int noTies;
    private int noLosses;
    private int noGoalsScored;
    private int noGoalsReceived;

    //Constructor
    Team(String n, Country c, Gender g){
        this.name = n;
        this.country = c;
        this.genderClass = g;
    }
```

*Destacar además que en el constructor no pasamos las estadísticas por parámetro pues al definirlas como atributos de tipo entero se inicializan a cero. Ésto se modificará a través de otros métodos.

Luego, se proporcionan métodos para acceder a información específica del equipo (los llamados Getters), como su nombre, país y género.

```java
//Getters
String getName(){
    return this.name;
}
Country getCountry(){
    return this.country;
}
Gender getGenderClass(){
    return this.genderClass;
}
```

También se han implementado métodos para jugar partidos *playMatch(...)*, actualizando así las las estadísticas del equipo y pudiendo imprimir imprimirlas con el método *printStats()*:

```
//Methods
void playMatch(int gs, int gr){
    if(gr>gs){
        this.noLosses = this.noLosses + 0;
    } else if(gr<gs){
        this.noWins = this.noWins + 3;
    } else {
        this.noTies = this.noTies + 1;
    }

    this.noGoalsScored += gs;
    this.noGoalsReceived += gr;
    this.noMatches++;
}

void printStats(){
    System.out.println("\n" + this.name + " statics");
    System.out.println("- noMatches: " + this.noMatches);
    System.out.println("- noWins: " + this.noWins);
    System.out.println("- noTies: " + this.noTies);
    System.out.println("- noLosses: " + this.noLosses);
    System.out.println("- noGoalsScored: " + this.noGoalsScored);
    System.out.println("- noGoalsReceived: " + this.noGoalsReceive

}
```

- **Player:**

Atributos: *female, name, age, nationality, noMatches, noTackles, noPasses, noShots, noAssists, y noGoals*.
Métodos: La clase Player consta de un constructor que inicializa a un jugador con atributos como su género, nombre, edad y nacionalidad. A continuación una parte del código implementado para esta clase:

```java
public class Player{
    //Atributos
    private boolean female;
    private String name;
    private int age;
    private Country nationality;
    private int noMatches;
    private int noTackles;
    private int noPasses;
    private int noShots;
    private int noAssists;
    private int noGoals;

    //Constructor
    Player(boolean g, String n, int a, Country nat ){
        female = g;
        name = n;
        age = a;
        nationality = nat;
    }
```

Se proporcionan métodos para acceder a información específica del jugador, como su género, edad, nombre y nacionalidad (Getters). Además, se implementan métodos para actualizar las estadísticas del jugador y para imprimir estas estadísticas:

```java
//Methods
void update(int t, int p, int s, int a, int g){
    this.noTackles += t;
    this.noPasses += p;
    this.noShots += s;
    this.noAssists += a;
    this.noGoals += g;
    this.noMatches++;
}

void printStats(){
    System.out.println("\n" + this.name + " stats");
    System.out.println("- noMatches: " + this.noMatches);
    System.out.println("- noTrackles: " + this.noTackles);
    System.out.println("- noPasses: " + this.noPasses);
    System.out.println("- noShots: " + this.noShots);
    System.out.println("- noAssists: " + this.noAssists);
    System.out.println("- noGoals: " + this.noGoals);

}
```

- **Country:**

Atributos: name.
Métodos: La clase Country contiene un constructor para inicializar un país con su nombre y un método para obtener el nombre del país.
* Dicha clase viene dada por la práctica.

2. *Descripción soluciones alternativas*

En esta práctica hemos planteado distintas soluciones alternativas, como por ejemplo:

- Utilizar como atributo *country* un tipo String, en vez de haber creado una nueva clase. La implementación escogida refleja las ventajas de la programación orientada a objetos, como la abstracción, la encapsulación y la mejora de la legibilidad y el mantenimiento del código. En resumen, esta decisión se basa en el diseño de software para lograr un código más organizado y extensible.

- Utilizar como atributo *genderClass* un tipo String o int, en vez de haber creado una enumeración. Me he decantado por dicha implementación pues el uso de un enumerador Gender en lugar de una cadena o número es una elección sólida en términos de diseño de código, ya que mejora la

legibilidad, la seguridad y la facilidad de mantenimiento, lo que es fundamental en la programación orientada a objetos.

3. *Conclusión*

La solución implementada funciona eficazmente para la gestión de equipos y jugadores de fútbol. El programa de prueba Test demuestra la funcionalidad de las clases al crear equipos, añadir y eliminar jugadores, simular partidos y mostrar estadísticas. Los equipos asignan correctamente a los jugadores según su género, cumpliendo con las reglas establecidas. Analizamos dicha clase Test con detenimiento:

En primer lugar creamos las instancias de las clases necesarias y añadimos jugadores a cada equipo.

```
public static void main(String[] args) {

    //Create country instances
    Country spain = new Country(n:"Spain");
    Country france = new Country(n:"France");


    //Create player instances
    Player p1 = new Player(g:false, n:"Cristiano Ronaldo", a:35, spain);
    Player p2 = new Player(g:false, n:"Lionel Messi", a:33, spain);
    Player p3 = new Player(g:false, n:"Sergio Ramos", a:34, spain);
    Player p4 = new Player(g:false, n:"Kylian Mbappé", a:21, france);
    Player p5 = new Player(g:false, n:"Neymar", a:28, france);
    Player p6 = new Player(g:false, n:"Karim Benzema", a:32, france);
    Player p7 = new Player(g:true, n:"Alexia Putellas", a:29, spain);

    //Create team instances
    Team t1 = new Team(n:"FC Barcelona", spain, Team.Gender.Male);
    Team t2 = new Team(n:"PSG", france, Team.Gender.Male);

    //Add players to the teams
    t1.addPlayer(p1);
    t1.addPlayer(p2);
    t1.addPlayer(p3);
    t2.addPlayer(p4);
    t2.addPlayer(p5);
    t2.addPlayer(p6);
```

Seguidamente, imprimimos por pantalla la información de cada equipo utilizando los métodos correspondientes:

```java
//Print players of each team
System.out.println(x:"==================================");
System.out.println(x:"Printing players of each team");
System.out.println(x:"==================================");
t1.printPlayers();
t2.printPlayers();

//Print teams information
System.out.println(x:"\n==================================");
System.out.println(x:"Printing info of each team");
System.out.println(x:"==================================");
//info team 1
System.out.println("Name of team 1: " + t1.getName());
System.out.println("Country of team 1: " + t1.getCountry().getName());
System.out.println("Gender of team 1: " + t1.getGenderClass());
//info team 2
System.out.println("\nName of team 2: " + t2.getName());
System.out.println("Country of team 2: " + t2.getCountry().getName());
System.out.println("Gender of team 2: " + t2.getGenderClass());
```

```
* Cristiano Ronaldo added to the team FC Barcelona

* Lionel Messi added to the team FC Barcelona

* Sergio Ramos added to the team FC Barcelona

* Kylian Mbappé added to the team PSG

* Neymar added to the team PSG

* Karim Benzema added to the team PSG
==================================
Printing players of each team
==================================

Printing players of FC Barcelona
- Cristiano Ronaldo
- Lionel Messi
- Sergio Ramos

Printing players of PSG
- Kylian Mbappé
- Neymar
- Karim Benzema

==================================
Printing info of each team
==================================
Name of team 1: FC Barcelona
Country of team 1: Spain
Gender of team 1: Male

Name of team 2: PSG
Country of team 2: France
Gender of team 2: Male
```

A continuación, jugamos determinados partidos actualizando así las estadísticas de los distintos equipos e imprimiendolas por pantalla. Actualizaremos también las estadísticas individuales de cada jugador y las imprimimos por pantalla.

```
//Play matches
System.out.println(x:"\n==================================");
System.out.println(x:"Printing stats of each team");
System.out.println(x:"==================================");
t1.playMatch(gs:2, gr:1);
t1.playMatch(gs:1, gr:1);
t1.playMatch(gs:0, gr:1);
t2.playMatch(gs:1, gr:1);
t2.playMatch(gs:1, gr:2);
t2.playMatch(gs:3, gr:1);

//Print stats of each team
t1.printStats();
t2.printStats();

//Update players stats
p1.update(t:0, p:0, s:3, a:0, g:1);
p2.update(t:0, p:0, s:2, a:0, g:0);
p3.update(t:0, p:0, s:0, a:0, g:0);
p4.update(t:0, p:0, s:1, a:0, g:0);
p5.update(t:0, p:0, s:0, a:0, g:0);
p6.update(t:0, p:0, s:0, a:0, g:0);

//Print players stats
System.out.println(x:"\n==================================");
System.out.println(x:"Printing stats of each player");
System.out.println(x:"==================================");
p1.printStats();
p2.printStats();
p3.printStats();
p4.printStats();
p5.printStats();
p6.printStats();
```

```
==================================
Printing stats of each team
==================================

FC Barcelona statics
- noMatches: 3
- noWins: 3
- noTies: 1
- noLosses: 0
- noGoalsScored: 3
- noGoalsReceived: 3

PSG statics
- noMatches: 3
- noWins: 3
- noTies: 1
- noLosses: 0
- noGoalsScored: 5
- noGoalsReceived: 4
```

```
==================================
Printing stats of each player
==================================

Cristiano Ronaldo stats
- noMatches: 1
- noTrackles: 0
- noPasses: 0
- noShots: 3
- noAssists: 0
- noGoals: 1

Lionel Messi stats
- noMatches: 1
- noTrackles: 0
- noPasses: 0
- noShots: 2
- noAssists: 0
- noGoals: 0
```

Además, probaremos a cambiar un jugador del equipo 1 al equipo 2. Una vez realizado dicho cambio, imprimimos por pantalla la lista actualizada de jugadores.

```java
//Update players of teams
t1.removePlayer(p1);
t2.removePlayer(p4);
t1.addPlayer(p4);
t2.addPlayer(p1);

//Print players of each team
System.out.println(x:"==================================");
System.out.println(x:"Printing updated players of each team");
System.out.println(x:"==================================");
t1.printPlayers();
t2.printPlayers();
```

```
* Cristiano Ronaldo removed of the team FC Barcelona

* Kylian Mbappé removed of the team PSG

* Kylian Mbappé added to the team FC Barcelona

* Cristiano Ronaldo added to the team PSG
==================================
Printing updated players of each team
==================================

Printing players of FC Barcelona
- Lionel Messi
- Sergio Ramos
- Kylian Mbappé

Printing players of PSG
- Neymar
- Karim Benzema
- Cristiano Ronaldo
```

Finalmente, crearemos una jugadora femenina e intentaremos añadirla a un equipo masculino, comprobando así que al no corresponder su género con el del equipo ésto no sería posible. Por último, crearemos un equipo mixto y la añadiremos, siendo esta vez posible por concordancia de género.

```java
//Add Female to team 1
System.out.println(x:"\n==================================");
System.out.println(x:"Try to add a girl to team 1");
System.out.println(x:"==================================");

System.out.println(x:"\nPrinting info of player 7");
System.out.println("- Name: " + p7.getName());
System.out.println("- Age: " + p7.getAge());
System.out.println("- Country: " + p7.getNationality().getName());
if(p7.isFemale() == true){
    System.out.println(x:"- Gender: Female" );
} else {
    System.out.println(x:"- Gender: Male");
}

t1.addPlayer(p7);
t1.printPlayers();

//Create a mixed team
Team t3 = new Team(n:"Mixed Team", spain, Team.Gender.Mixed);
t3.addPlayer(p1);
t3.addPlayer(p2);
System.out.println(x:"\n==================================");
System.out.println(x:"Trying to add a girl to mixed team");
System.out.println(x:"==================================");

t3.addPlayer(p7);
t3.printPlayers();
```

```
================================
Try to add a girl to team 1
================================

Printing info of player 7
- Name: Alexia Putellas
- Age: 29
- Country: Spain
- Gender: Female

* Alexia Putellas can't be added to the team FC Barcelonadue to gender issues

Printing players of FC Barcelona
- Lionel Messi
- Sergio Ramos
- Kylian Mbappé

* Cristiano Ronaldo added to the team Mixed Team

* Lionel Messi added to the team Mixed Team

================================
Trying to add a girl to mixed team
================================

* Alexia Putellas added to the team Mixed Team

Printing players of Mixed Team
- Cristiano Ronaldo
- Lionel Messi
- Alexia Putellas
```

En resumen, la solución propuesta satisface las necesidades de gestión de equipos y jugadores de manera efectiva y ejemplifica dicha aplicación a partir de la clase test a la perfección.