INFORME LABS 2 POO

1. ¿Qué debe hacer el programa y cuáles son las clases y métodos que debes definir?

Para la práctica el objetivo era empezar a añadir equipos a ligas y simular partidos siguiendo las clases creadas en el seminario. Seguidamente el programa debe hacer un print de los equipos que están en la liga, crear un calendario con los partidos y simularlos imprimiendo el resultado. Lo primero que hemos hecho ha sido crear 2 archivos nuevos, por las 2 clases nuevas league y match.

La primera clase ha sido **league**, cuyos atributos són name, country, gender, teams, numMatches y matches. El constructor consta del nombre, country y género por inicializar. En cuanto a atributos tenemos los getters (ya vistos en todas las otras clases).
→ **addTeam()**: que comprueba el género del equipo que corresponda con el de la liga.
→ **generateMatches():** que genera todos los partidos de la liga ida y vuelta.
→ **printRounds():** que imprimiría los partidos a jugar en la liga.
→ **simulateMatches():** llama a la simulación de partidos generados por la liga dando resultado aleatorio, goleadores y pases de goles simulados de la misma manera.
→ **printMatches():** imprime la suma de partidos de la liga con resultado y goleadores.
→ simulateSingleMatches que simula solo un partido no toda la liga y printSingleMatches.

La segunda clase ha sido **match** con atributos homeTeam, awayTeam, homeGoals, awayGoals, lista de homeScorers y awayScorers. Su constructor introduce homeTeam y awayTeam, con esto ya se disputa un partido. El método principal es el de simulateMatch este genera un resultado aleatorio al asignar goles y goleadores a los equipos involucrados. Utiliza números aleatorios para determinar la cantidad de goles y selecciona jugadores para marcar goles y asistencias. Luego, actualiza las estadísticas de los jugadores en función de su desempeño en el partido. Los demás métodos són los getters de las estadísticas que se mueven en el simulate cómo goleadores locales o asistentes, finalmente tenemos un printMatch que lo imprime todo.

Finalmente en el main creamos jugadores, países, equipos y ligas pasando directamente los atributos del constructor manualmente.

Tras inicializar los objetos llamamos a los métodos de league para crear los partidos simularlos y imprimir las estadísticas para comprobar que todo funciona correctamente.

A parte de estas tres clases, manteniamos las clases hechas en el Laboratorio 1, Player, Team y Country que recibieron alguna modificación en algun metodo.

Player: El updateStats de cada jugador recibe una variación, ahora en vez de recibir los datos del main, este método los asigna aleatoriamente como vemos a continuación:

```
public void updateStatsPlayers(){
        Random random = new Random();
        this.matchesPlayed++;
```

```
        this.shots += random.nextInt(7); // Genera un número aleatorio de goles para el equipo
local (0-6)
        this.tackles += random.nextInt(20);
        this.passes += random.nextInt(50);
    }
```

Team: En la clase Team, tambien hemos modificado el updateStats de cada equipo, en el Laboratorio 1, recibia los goles anotados por cada equipo y la función almacenaba a la clase sus atributos. Ahora, con la clase Match, simulada a cada partido y con atributos al equipo local y visitante ya podemos actualizar a cada equipo así:

```
public void updateStats(Match m){
        if (this.name == m.getHomeTeam().getName()) {
            this.matchesPlayed++;
            this.goalsScored += m.getHomeGoals();
            this.goalsAgainst += m.getAwayGoals();
            if (m.getHomeGoals()>m.getAwayGoals()){
                this.points += 3;
                this.wins++;
            } else if (m.getHomeGoals()< m.getAwayGoals()){
                this.losses++;
            } else{
                this.points++;
                this.ties++;
            }
        }
        else if (this.name == m.getAwayTeam().getName()) {
            this.matchesPlayed++;
            this.goalsScored += m.getAwayGoals();
            this.goalsAgainst += m.getHomeGoals();
            if (m.getHomeGoals() < m.getAwayGoals()){
                this.points += 3;
                this.wins++;
            } else if (m.getHomeGoals()> m.getAwayGoals()){
                this.losses++;
            } else{
                this.points++;
                this.ties++;
            }
        } else{
            System.out.println("El partido llevado a estadísticas no existe.");
        }
    }
```

Country se mantuvo igual.

2. ¿Qué soluciones alternativas se consideraron y por qué se eligió la solución actual?

Una de las opciones que te daba el laboratorio era como implementar las estadisticas de los jugadores en cada partido. Durante la elaboración del código se planteó la manera de hacerlas todas aleatorias, eliminarlas o representar las estadísticas según su posición en el campo. Finalmente, se añadieron las estadísticas aleatoriamente menos las de gol y asistencia. La práctica nos planteaba clara la opción de que con la lista de goleadores sea más fácil añadir la estadística directamente al Player, pero no nos parecía lógico colocar las asistencias aleatorio porque en general, si hay tantos goles, hay tantas asistencias. La implementación fue la siguiente. (un trozo del simulate):

```
// Simulate away team's goals and goal scorers
        for (int i = 0; i < awayGoals && !awayTeam.getPlayers().isEmpty(); i++) {
            randomGoalIndex = random.nextInt(awayTeam.getPlayers().size());
            Player scorer = awayTeam.getPlayers().get(randomGoalIndex);
            awayScorers.add(scorer);
            scorer.marcarGol(); // Actualiza estadísticas del jugador
            randomAsistIndex = random.nextInt(awayTeam.getPlayers().size());
            Player asistPlayer = awayTeam.getPlayers().get(randomAsistIndex);
            while(scorer.getName() == asistPlayer.getName()){
                newrandomAsistIndex = random.nextInt(awayTeam.getPlayers().size());
                asistPlayer = awayTeam.getPlayers().get(newrandomAsistIndex);
            }
            asistPlayer.asistir();
        }
```

Funciones en el Player:

```
public void marcarGol() {
      this.goals++;
   }

   public void asistir(){
      this.assists++;
   }
```

Como última solución alternativa pero esta es más estética es la manera de imprimir los partidos donde añadimos a parte del resultado, la lista de goleadores haciendo referencia a su equipo.

```
Atlético de Madrid 1 - 1 Real Madrid
-------------------------------------
Crónica:
Gol Atlético de Madrid--> Herrera
Gol Real Madrid--> Benzema
```

3. ¿Cómo funcionó la solución en la práctica? ¿Hubo dificultades en la implementación o dudas?

Uno de los problemas que hemos tenido a la hora de la implementación ha sido a la hora de añadir equipos a League. Intentábamos con el método de addTeam() pero nos daba error. El equipo debía de añadirse en una linkedlist pero no lo hacía. Finalmente localizamos el error que era que no inicializamos la lista en el constructor de liga. Al ver este error hemos corregido otro que hubiésemos tenido más adelante que era el de no inicializar tampoco la linkedlist de partidos. Así que creamos las dos listas vacías y el equipo se añadió correctamente.

Por lo demás, y con tiempo para ir haciendo pruebas se ha conseguido el Laboratorio 2 sin mucha más dificultad.