Lab2

INTRODUCCIÓN

El segundo laboratorio ha sido, sin duda, un reto mayor que el que se nos presento el en primer laboratorio. Hemos tenido que trabajar mucho mas y ayudarnos entre nosotros mucho mas para resolver los problemas que nos hemos enfrentado esta vez. En el laboratorio debíamos hacer una extensión o actualización de nuestro anterior programa. Esta vez debíamos hacer una ampliación y ser capaces de crear una liga entre n partidos en los que se darían $(n,2)=n!/2!\cdot(n-2)!$ partidos en total.

Esta vez hemos desarrollado dos nuevos objectos e implementado nuevos métodos a otros objectos ya programados.

En primer lugar tenemos el objecto “Match”, este objecto se encargará de todo lo que involucre la realización del partido. En él, declaramos seis atributos, los primeros y mas esenciales son dos objectos “Team”, nombrados “homeTeam” y “awayTeam”. En segundo lugar tenemos los goles dados por cada equipo declarados de forma sistemática como “awayGoals” y “homeGoals” en tipo entero. Finalmente también declaramos dos listas de tipo “Player” las cuales cada una contendrá la lista de jugadores que han marcado de cada equipo, estas dos variables son “homeScorers” y “awayScorers”. El objecto “Match” contiene cuatro métodos relativamente simples, el primero es el constructor, “Match (h: Team, a: Team)”. El segundo lo podríamos definir como los métodos “getters”. En tercer lugar tenemos el principal y mas importante, el “”simulateMatch”, que pretende simular un partido entre el equipo local y el visitante, generando un numero ‘random’ de goles para cada equipo y crear una lista aleatoria de jugadores que hayan marcado gol. Y en ultimo lugar el printMatch () el cual se encargará de imprimir en consola los datos del partido (resultado y lista de jugadores que han marcado).

El segundo objecto será la estructura de la liga en si. Esta se a declarado como “League” y contiene cinco atributos básicos como el name (“String”), un atributo de tipo “Country” declarado como country, el atributo gender y para finalizar dos listas, una de tipo “Match” y otra de tipo “Team”. La primera se encargará de contener todos los partidos que se darán en la liga, y la segunda se encargará de contener toda la lista de equipos que jugaran en la liga. El objeto de liga necesita atributos que le permitan llamar al objecto “Match” para así poder empezar cada partido y ser capaz de estructurar todos los partidos que se deberán jugar. La función principal para llegar a cabo la liga es la función “generateMatches” la cual se encargará de crear la lista de partidos que se deberán realizar. En segundo lugar, el método “simulateMatches()” el cual será la que llamará a la función “simulateMatch” del objeto “Match” para realizar todos los partidos, declarar un ganador y actualizar las estadísticas. En tercer lugar, los métodos “printMatches” y “printLeagueClasification” que imprimen en la consola los resultados de todos los partidos y la clasificación final. En último lugar se han definido el conjunto de todos los métodos “getters” y el método “addTeam (t: Team)” el cual permite añadir equipos a la lista de equipos.

En último lugar exponemos las actualizaciones realizadas en otros objectos ya declarados en el anterior laboratorio. En este caso, hemos actualizado el objecto "Player" con el nuevo método "update" y su implementación, el cual nos permitirá actualizar las estadísticas de cada jugador y un nuevo método "getPlayers". Y hemos actualizado el método "updateStats()" de la clase "Team" para actualizar las estadísticas de cada jugador.

METODOLOGÍA

Esta vez ha sido mas fácil plantear el nuevo laboratorio, ya que las bases del programa estaban echas para el Lab1 y ahora solo era aplicar lo aprendido anteriormente y mejorarlo con mas clases, además de ampliar el programa. Entonces, durante el desarrollo del segundo laboratorio no ha habido muchos problemas o inconvenientes.

De todas formas, el mayor reto ha sido la implementación del método "updateStats()" de la clase "Team" para gestionar correctamente las estadísticas de los jugadores. Donde se precisa contar el numero de goles marcados de cada partido por un jugador. El problema fue, que en vez de hacer un contador para ver cuantas veces estaba el nombre de un jugador en la lista de goleadores, directamente cogíamos el valor del número de goles de todos los partidos y además se lo poníamos a todos los jugadores, no sólo uno en particular. Por lo tanto, todos los jugadores tenían el número total de goles del equipo.

Para solucionarlo hemos echo un bucle "for" para cada uno de los jugadores de la lista de jugadores dependiendo si es local o visitante y un vez dentro generamos una variable "int" con valor cero para contar el numero de goles que haya echo un jugador. Después, otro bucle "for" y recorrer cada jugador con la lista de goleadores del partido para ver si el nombre coincide y sumar uno a la variable "goalsScored". Así, se suman correctamente los goles de cada jugador independientemente de la cantidad de goles echos en un solo partido (ya que antes también solo apuntaba un gol por partido). En consecuencia, ya se puede hacer el "update" de todos los jugadores.

```
for (Player player : playerList) {
    int goalsScored = 0;
    for (Player scorer : match.getHomeScorers()) {
        if (scorer == player) {
            goalsScored++;
        }
    }
    player.update(tackles:0, passes:0, shots:0, assists:0, goalsScored);
}
```

Por otro lado, otro problema que tuvimos fue que la función “Random”, nos daba el siguiente error: *Exception in thread "main" java.lang.IllegalArgumentException: bound must be positive:*

at java.base/java.util.Random.nextInt(Random.java:557)

at Lab2.Match.simulateMatch(Match.java:60).

Este problema sucede cuando el intervalo dado a la función “random” para generar un numero aleatorio es negativo. Pero nosotros teníamos un valor positivo (4), por lo que este no era nuestro error. Probamos diferentes soluciones, hasta que nos dimos cuenta que algunos equipos no estaban formados por suficientes jugadores. Entonces, añadimos los jugadores necesarios y todo funcionó correctamente.

En general, el código logró cumplir con los objetivos planteados inicialmente y se utilizó la clase "FootballAplication" para probar y demostrar las implementaciones realizadas.

CONCLUSIÓN

Una vez terminado el laboratorio podemos afirmar que la implementación del nuevo código con las nuevas clases y métodos, funcionan correctamente y han ampliado de una forma mas extensa el funcionamiento de las clases y los objetos mediante la simulación de una liga.

Aunque se encontraron desafíos durante el laboratorio, gracias a la clase “FootballAplication” donde hemos ido haciendo pruebas y testeos mediante la consola, todo se ha corregido. Y al final, el programa funciona tal y como se espera, donde se gestionan eficazmente equipos y jugadores de fútbol.

En conclusión, se ha cumplido el objetivo y se ha comprendido la importancia de diseñar un programa bien planificado.