PRACTICA GRUPO

LIGA DE FUTBOL

Carlos Soler Garcia

Roberto García Sánchez

Adrián López Espejo

Juan Antonio Enríquez Bustos

Clase Principal

La clase principal es donde se generan todos los Jugadores, Equipos, Entrenadores y Árbitros que componen la Liga, además de generar precisamente dicha liga y hacer un llamamiento a la clase menú, que es el objeto con el que interactúa el usuario para pedir los datos que desee.

Los jugadores, equipos, entrenadores y árbitros se generan de manera aleatoria en sus métodos, los cuales devuelven una lista de Jugadores, una lista de Equipos, una lista de árbitros y un solo Entrenador que se le asigna a cada equipo que se le introduce al método.

Principal.java

```
52
53              Partido [] partidos = jornadas[i].getPartidos();
54
55              for (Partido par: partidos) {
56                  int golesLocales = (int) Math.floor(Math.random()*MAXIMOGOLES);
57                  int golesVisitantes = (int) Math.floor(Math.random()*MAXIMOGOLES);
58                  par.setgLocal(golesLocales);
59                  par.setgVisitante(golesVisitantes);
60                  System.out.println(par);
61              }
62              jornadas[i].terminar();
63          }
64
65      }
66
67      private static Jugador[] crearJugadores(int numeroJugadores, int edad, Equipo equipo) {
68          //Listado de Nombres, Apellidos, Posiciones para generador random
69          String[] nombres = {"Antonio", "Pepito", "Alejandra", "Ismael", "Hugo", "Oliver","Kalesi",
70                  "Ingrid","Astrid","Indira","Jenny","Jessi","Vane","Joel","Bruno",
71                  "Sasha","Billie","Masha","Pingu"};
72          String[] apellidos = {"Messi", "Vinicius", "Cristiano", "Ronaldo", "PiquÃ©","Bale (lesionado)",
73                  "Amunike","N'kono","ButragueÃ±o","SanchÃ-s","Neymar","Batistuta","Maradona",
74                  "PelÃ©","Beckenbauer"};
75          String[] posiciones = {"Portero/a","Defensa","Centrocampista","Delantero/a"};
76
77          //Estructura de Array de Jugadores
78          Jugador[] jugadores = new Jugador[numeroJugadores];
79
80          for (int i=0; i<numeroJugadores; i++) {
81              //Crear un Jugador
82              Jugador jug = new Jugador();
83              //Nombre
84              int numero = (int) Math.floor(Math.random()*nombres.length);
85              String nombre = nombres[numero];
86              jug.setNombre(nombre);
87
88              //Apellidos
```

Clase Liga

La clase Liga básicamente es donde se alojan todos los equipos, el Calendario (con sus jornadas y partidos), los árbitros y la clasificación. El constructor de Liga exige dos parámetros para poder generar el objeto, uno es la lista de Equipos y el otro la lista de Árbitros, los cuales se pasan como argumentos a los atributos de la liga antes de generar el Calendario, ya que este requiere de una lista de Equipos también. El atributo Nombre viene por defecto como "Liga Sin Nombre", ya que la idea era añadir una opción en el menú para que el usuario pudiese cambiar el nombre de la liga como quisiera.

```
1
2  public class Liga {
3
4      private String nombre;
5      private Equipo[] equipos;
6      private Calendario calendario;
7      private Arbitro[] arbitros;
8      private Clasificacion clasificacion;
9
10
11     public Liga (String nombre, Equipo[] equipos, Arbitro[] arbitros) {
12         this.nombre=nombre;
13         this.equipos=equipos;
14         this.arbitros=arbitros;
15         this.calendario=new Calendario(this.equipos,this.arbitros);
16     }
17
18     public String getNombre() {
19         return nombre;
20     }
21     public void setNombre(String nombre) {
22         this.nombre = nombre;
23     }
24     public Equipo[] getEquipos() {
```

Clase Calendario

Esta clase es una de las que tiene más complejidad, ya que es la encargada de recibir los equipos desde liga y emparejarlos para asignar los partidos de cada jornada teniendo en cuenta una serie de condiciones, que debe de haber ida y vuelta, que no pueden repetirse partidos y que si los equipos son impares uno tiene que ser el que descansa.

El único atributo de esta clase es un array de Jornadas y el constructor requiere una lista de equipos. Este constructor crea la longitud del array de Jornadas teniendo en cuenta si son equipos pares o impares y llama al método emparejamiento que retorna dicho array de Jornadas.

Método emparejamiento:

El algoritmo de emparejamiento intenta que los equipos jueguen tanto en casa como de visitante de manera alternativa. El patrón que he seguido es el siguiente:

Jornada 1		
E1 ↓	-	E2 =
E3 ↓	-	E4 =
E5 ↓	-	E6 =
EC ↑↑	-	E7 =

Jornada 2		
EC↓↓	-	←E2
E1 =	-	E4↑
E3 =	-	E6↑
E5 →	-	E7↑

En la jornada 1 realmente no se mueven los equipos, se muestran tal cual vienen en la lista de Equipos que se generó al inicio, en las jornadas pares es cuando se lleva a cabo el movimiento del primer cuadrante en el que solo se mueven los equipos que jugaron de locales, y en las jornadas impares se lleva a cabo el movimiento del segundo cuadrante.

La simbología es la siguiente, una flecha indica que se mueve una posición, dos flechas que se mueve hasta el lado opuesto de la tabla, y un igual que no se cambió de posición. Nosotros por motivos de hacer el código más simple decidimos que el movimiento uno cambiase de posición en lugar de los locales los visitantes.

Solo hay una pega con este algoritmo y es que el último equipo de la lista siempre juega de visitante porque no rota hacia los equipos locales, se podría modificar, pero llevaría más tiempo de implementación.

El método de emparejamiento a diferencia de cómo podría pensar alguno que devolvería una matriz con los equipos.

Clase Jornada

El constructor de la clase Jornada pide como argumentos de entrada una lista de equipos que es la que viene dada del método emparejamiento del Calendario.

```
1  import java.util.Arrays;
2
3  public class Jornada {
4
5      private Partido[] partidos;
6      private boolean terminada=false;
7
8      public Partido[] getPartidos() {
9          return partidos;
10     }
11
12     public void setPartidos(Partido[] partidos) {
13         this.partidos = partidos;
14     }
15
16
17     public boolean isTerminada() {
18         return terminada;
19     }
20
21     public void setTerminada(boolean terminada) {
22         this.terminada = terminada;
23     }
24
```

Como expliqué la lista de equipos emparejados no es una matriz sino un simple array ordenado como EquipoLocal, EquipoVisitante, EquipoLocal, EquipoVisitante, EquipoLocal, EquipoVisitante… etc. De esta manera solo hay que coger las posiciones del array de dos en dos y a la hora de crear el objeto partido el primer equipo se asignará como equipo local y el segundo como equipo visitante.

¿Y qué ocurre si los equipos son impares? Realmente no hay mucho cambio, el ejemplo sería el mismo, pero a la hora de asignar equipo local y equipo visitante al partido si uno de los dos es un espacio nulo del array simplemente ese partido no se genera y el equipo de esa pareja es el que descansa ya que no tiene enfrentamiento.

Como se muestra en la captura el método generarPartido() crea un array de partidos. Un for que recorre el array equiposEmparejados (que viene de la clase Calendario) de dos en dos, como se ha explicado antes, y con el condicional if comprobamos si equipo local o equipo visitante es null antes de generar el objeto partido y asignarle los valores.

Clase Partido

La clase partido tiene los atributos Equipo Local, Equipo Visitante, Arbitro y los goles de cada equipo. Además tiene los métodos que pueden ser llamados de goles randoms para equipos locales y equipos visitantes.

```java

public class Partido {

    private Equipo local;
    private Equipo visitante;
    private int gLocal;
    private int gVisitante;
    private Arbitro arbitro;

    public Equipo getLocal() {
        return local;
    }
    public void setLocal(Equipo local) {
        this.local = local;
    }
    public Equipo getVisitante() {
        return visitante;
    }
    public void setVisitante(Equipo visitante) {
        this.visitante = visitante;
    }
    public int getgLocal() {
        return gLocal;
    }
    public void setgLocal(int gLocal) {
        this.gLocal = gLocal;
    }
    public int getgVisitante() {
        return gVisitante;
    }
```

Clase Menú

La clase menú como constructor tiene de entrada el objeto liga. El menú imprime las opciones que el usuario ve con el método imprimirMenu(), le pide una opción al usuario desde el scanner y según lo que este escoja se llamará a otro método gracias a un Switch. Los métodos que llama el Switch son específicos de esta clase y se encuentran en la parte posterior del código.

```
58    public static void imprimirMenu() {
59        System.out.println("\n*************************************");
60        System.out.println("********** Menu Principal ***********");
61        System.out.println("* 1.- Ver la clasificacion actual     *");
62        System.out.println("* 2.- Ver el calendario               *");
63        System.out.println("* 3.- Introducir nuevos resultados    *");
64        System.out.println("* 4.- Mostrar la lista de equipos     *");
65        System.out.println("* 5.- Mostrar la lista de arbitros    *");
66        System.out.println("* 6.- Salir de la aplicacion          *");
67        System.out.println("*************************************");
68        System.out.print("Introduzca una opcion: ");
69    }
70
```

Clase Persona

Se trata de una superclase que guarda atributos que son comunes a otras subclases, jugador, entrenador y arbitro. Aquí encontramos atributos como la edad, nombre y apellidos. Es una clase bastante sencilla y sirve como base para las otras clases hijas.

```
2  public abstract class Persona {
3      private String nombre;
4      private String apellidos;
5      private int edad;
6
7      public String getNombre() {
8          return nombre;
9      }
10     public void setNombre(String nombre) {
11         this.nombre = nombre;
12     }
13     public String getApellidos() {
14         return apellidos;
15     }
16     public void setApellidos(String apellidos) {
17         this.apellidos = apellidos;
18     }
19     public int getEdad() {
20         return edad;
21     }
22     public void setEdad(int edad) {
23         this.edad = edad;
24     }
25
```

Clase Jugador

Es una clase que extiende persona y en ella se almacenan los atributos referentes al jugador, dorsal, posición, categoría en la que juega y equipo al que pertenece. Es una clase bastante importante ya que son los componentes esenciales de los equipos.

```
public final class Jugador extends Persona{
    private String categoria;
    private String posicion;
    private int dorsal;
    private Equipo equipo;

    @Override
    public void setEdad(int edad) {
        super.setEdad(edad);
        categoria=setCategoria(edad);

    }

    public String getCategoria() {
        return categoria;
    }
    private String setCategoria(int edad) {
        switch(edad) {
        case 4:
        case 5:
```

Clase Entrenador

Es una clase que viene de la super clase Persona, tiene solo 2 atributos que son el número de licencia y el equipo al que pertenecen, no es una clase muy importante.

```
public class Entrenador extends Persona{

    private int numeroLicencia;
    private Equipo equipo;

    public int getNumeroLicencia() {
        return numeroLicencia;
    }
    public void setNumeroLicencia(int numeroLicencia) {
        this.numeroLicencia = numeroLicencia;
    }
    public Equipo getEquipo() {
        return equipo;
    }
    public void setEquipo(Equipo equipo) {
        this.equipo = equipo;
    }
```

Clase Árbitro

Como las anteriores es una clase hija de la clase Persona, es una clase bastante importante en el programa ya que tiene que existir tantos árbitros como partidos se jueguen en la jornada, a cada uno se le asignará un partido que pitar cada jornada. Aunque solo tengo un atributo el papel que desempeña al pitar cada jornada un partido es importante y tiene que quedar constancia de ello.

```
public class Arbitro extends Persona {
    int licencia;

    public int getLicencia() {
        return licencia;
    }

    public void setLicencia(int licencia) {
        this.licencia = licencia;
    }

    @Override
    public String toString() {
        return "Arbitro [licencia=" + licencia + ", getNombre()=" + getNom
                + ", getEdad()=" + getEdad() + "]";
    }


}
```

Clase Clasificación

En esta clase se revisan todas las jornadas jugadas y se hace un cómputo de los puntos y estadísticas de cada equipo, acto se guido se realiza una tabla donde se ordenan, mediante el algoritmo Bubble Sort, los equipos, estando en primera posición aquellos que hayan marcado más goles. Esta clase también actualiza los partidos y sus resultados en la clase EquipoClasificacion.

```
1
2  public class Clasificacion {
3
4      private EquipoClasificacion[] tabla;
5
6      public Clasificacion(Equipo[] equipos, Calendario calendario) {
7          int numeroEquipos=equipos.length;
8
9          this.tabla = new EquipoClasificacion[numeroEquipos];
10         //Crear la tabla de clasificacion
11         for (int i=0;i<numeroEquipos;i++) {
12             this.tabla[i] = new EquipoClasificacion();
13             this.tabla[i].setEquipo(equipos[i]);
14         }
15
16         //Tenemos que cambiar el numero de jornadas a jugar
17
18         //Rellenamos la tabla de clasificacion
19
20         Jornada [] jornadas = calendario.getJornadas();
21
22         for (Jornada jor: jornadas) {
23             if (jor.isTerminada()) {
24
25
26                 Partido [] partidos = jor.getPartidos();
27                 for (Partido par: partidos) {
28                     //Quien es el local
29                     Equipo local = par.getLocal();
30                     //Quien es el visitante
31                     Equipo visitante = par.getVisitante();
32                     //Buscarlos en la clasificaciÃ³n
33                     int contador=0;
34                     EquipoClasificacion localClas = tabla[contador];
35                     while (localClas.getEquipo()!=local) {
36                         contador++;
37                         localClas=tabla[contador];
38                     }
```

Clase EquipoClasificacion

Esta clase guarda los numerosos atributos referentes a la clasificación de los equipos, para cada equipo guarda los partidos jugados, su resultado, y los goles a favor y en contra. Mediante esta clase podemos luego calcular la posición que tendría el equipo respecto a los otros simplemente revisando sus atributos

```
1
2  public class EquipoClasificacion {
3      private Equipo equipo;
4      private int jugados=0;
5      private int ganados=0;
6      private int empatados=0;
7      private int perdidos=0;
8      private int gFavor=0;
9      private int gContra=0;
.0     private int difGoles=0;
.1     private int puntos=0;
.2
.3
.4     public Equipo getEquipo() {
.5         return equipo;
.6     }
.7     public void setEquipo(Equipo equipo) {
.8         this.equipo = equipo;
.9     }
!0     public int getJugados() {
!1         return jugados;
!2     }
!3     public void setJugados(int jugados) {
!4         this.jugados = jugados;
!5     }
!6     public int getGanados() {
!7         return ganados;
```

Enlace donde se explica el detalladamente el sistema de emparejamientos.

https://www.cihefe.es/cuadernosdefutbol/2013/03/la-confeccion-del-calendario-de-liga/