UNIVERSIDADE da MADEIRA

Relatório do 2º projeto de POO
Futebol Manager

Professores:

Luis Ferreira

Yuri Almeida

Sergi Bermúdez

Alunos:

Paulo Alexandre Rodrigues Alves n.º 2120722

Renato Gabriel Silva Pêssego n.º 2121922

João Tomás Correia Abreu nº 2118722

Funchal
26 Dezembro 2023

Introdução

O projeto que elaboramos tem como objetivo o desenvolvimento de um jogo de gestão de equipas de futebol, oferecendo funcionalidades para inserção e consulta de informações relevantes. Este projeto que é baseado no jogo football manager abrange elementos cruciais do mundo do futebol, como jogadores, treinadores, partidas e ligas. A riqueza de detalhes sobre jogadores, incluindo estatísticas de desempenho, histórico de lesões e níveis de agressividade, contribui para uma representação de dados do ambiente desportivo.

Ao longo deste relatório, serão detalhadas as funcionalidades principais deste projeto, desde o acesso às informações de jogadores, treinadores e equipas, até a criação e análise estatística de partidas. A abordagem cuidadosa das situações de erro e validações contribui para a robustez e integridade do sistema, garantindo uma experiência consistente e livre de inconsistências ,tendo dados reais de equipas , jogadores e treinadores , proporcionando uma experiência mais próxima da realidade para quem jogar este jogo.

Desenvolvimento

Neste projeto que tinha como objetivo de criarmos o jogo football manager , começamos por criar as classes Arbitro.java , Equipa.java ,Ataque.java, Defesa.java , Jogador.java , Leitura.java ,Lesoes.java , Liga.java , Menu.java , Partida.java , Pessoa.java , SubMenu.java , Treinador.java , TreinadorAssistente.java , TreinadorPrincipal.java, PooProjetoFase2.java em que escrevemos os atributos nas classes e os seus métodos respectivos de cada classe. Na classe Jogador.java , cada jogador era identificado pelo nome, idade, posição, histórico de lesões, estatísticas de desempenho (ataque e defesa) e nível de agressividade. Depois criamos um método para que quanto maior o nível de agressividade que um jogador tem, maior era a probabilidade de ter um cartão amarelo ou vermelho. Depois criamos um método caso levasse cartão vermelho saia do jogo e as probabilidades de a equipa vencer eram menores.

Na classe da Equipa.java fizemos métodos para adicionarmos jogadores à equipa, garantindo que não ultrapasse o limite de 11 jogadores , removermos jogadores da equipa e verificarmos se um jogador já pertence à equipa.Atualizamos as estatísticas das equipas como número de vitórias, derrotas, empates, golos marcados e sofridos e calculamos o desempenho médio de cada equipa com base nos resultados dos jogos. Conseguimos

associar treinadores à equipa, incluindo treinador principal e assistente e definimos a tática da equipa com base na posição dos jogadores. Temos um método em que é para treino dos jogadores, melhorando o seu overall com base na preferência de posição do treinador assistente e numa chance de melhoria. Também conseguimos mudar a moral do treinador com base na sua preferência tática e overall.Em cada temporada remetemos as estatísticas de uma equipa.Quando é possível verificamos e atualizamos o estado dos jogadores indisponíveis para depois os integrarmos na equipa .

A classe Arbitro.java é responsável por representar um árbitro num jogo de futebol. Herda características básicas de uma pessoa, como o nome, esta classe também possui atributos específicos para a idade e experiência do árbitro.

O principal método implementado na classe é lancarMoedaAoAr, que simula o lançamento de uma moeda ao ar para determinar qual equipa (casa ou fora) terá a primeira posse de bola no jogo. O método gera aleatoriamente dois valores (0 ou 1) para representar as opções "cara" e "coroa". A escolha da equipa é determinada pelo resultado do lançamento da moeda, proporcionando um método aleatório e imparcial para iniciar a partida.

A classe Ataque.java representa as estatísticas relacionadas ao desempenho ofensivo do jogador. Contém atributos que registram o número de golos marcados, assistências realizadas, passes certos e dribles bem-sucedidos. Além disso, a classe inclui um método chamado mostrarInfoAtaque(), que exibe as informações de ataque, como golos, assistências, passes certos e dribles certos.

A classe Defesa.java representa as estatísticas relacionadas ao desempenho defensivo de um jogador. Possui atributos que registram informações essenciais para avaliar a eficácia do jogador na defesa durante as partidas. Os atributos que temos incluem o número de carrinhos certos, cartões vermelhos, cartões amarelos e bolas recuperadas, cortes certos e golos impedidos . Incluimos um método chamado mostrarInfoDefesa(), que exibe as informações de defesa,que mencionamos anteriormente.

Na classe Jogador.java que representamos um jogador de futebol em que inclui atributos como nome, idade, posição, equipa, histórico de lesões, informações de ataque e defesa, nível de agressividade e dias indisponíveis. Depois fizemos métodos para manipularmos e obtermos informações relacionadas com esses atributos. Esta classe encapsula informações relevantes para a representação de um jogador de futebol, facilitando a gestão e análise do seu desempenho, histórico de lesões e outras estatísticas associadas ao jogador.

Na classe Leitura.java criamos métodos em que liam as linhas dos ficheiros txt e a informação específica de cada ficheiro como estatísticas, treinadores principais, treinadores assistentes e jogadores. As informações são armazenadas em objetos correspondentes, como Equipa, TreinadorPrincipal, TreinadorAssistente e Jogador.

As informações sobre treinadores são lidas, realizadas e registradas, incluindo especializações e preferências táticas. Depois para cada equipa lê a informação sobre os 11 jogadores , depois faz a validação e adiciona-os à lista de jogadores da equipa. O código verifica se uma liga já existe antes de adicioná-la à lista de ligas. Se a liga existir, a equipa é adicionada à liga existente; caso contrário, uma nova liga é criada. Criamos ficheiros txt um para cada equipa de cada liga em que contêm as informações para serem lidas nesta classe.

Na classe Lesoes.java criamos métodos em que armazenam e manipulam informações sobre lesões e depois temos um método que exibe o tipo de lesão e a data da lesão.

A classe Liga.java irá representar uma liga de futebol , que contém atributos como o nome da liga, a jornada atual, listas de equipas, árbitros e partidas e o ano da temporada. Os métodos que utilizamos incluem funcionalidades para atualizar a jornada, iniciar uma nova temporada, aumentar as idades dos participantes, criar partidas, verificar confrontos entre equipas e também para saber quem jogou em casa no primeiro confronto entre duas equipas . Depois criamos um método toString que retorna o nome da liga com o número de jornadas e o número de equipas.

Na classe Menu.java criamos um menu que irá aparecer para o utilizador com diversas opções para simular jogos ,associar equipas a ligas , verificar a tabela de uma liga , estatísticas da equipa , aceder a informações de jogadores , treinadores e equipas . Usamos um verificação para ver se o utilizador escolhe uma opção válida e também para cada uma das opções escolhidas métodos para verificar e depois exibir os resultados pretendidos pelo utilizador , caso esses existam.

A classe Partida.java foi onde pusemos os métodos todos que permitissem a simulação de uma partida. O método golos na classe Partida simula a ocorrência de gols durante a partida com base em probabilidades que foram calculadas a partir das características dos jogadores. Os métodos probabilidade_marcar_golo e probabilidade_defender_golo calculam as chances de um jogador marcar ou defender um gol, dependendo da posição e habilidades desse jogador.O método atualizar_infos_equipas atualiza as estatísticas das equipes com base nos gols marcados e sofridos durante a partida.Usamos métodos para calcularmos as estatísticas defensivas e ofensivas individuais dos jogadores. Métodos como demasiados_cartoes_casa, demasiados_cartoes_fora e lesoes tratam de situações como cartões e lesões durante o jogo.O método executarpartida coordena a execução da partida, considerando condições como expulsões e lesões. O método setresultado define o resultado da partida.

A classe Pessoa.java irá representar as informações sobre um jogador que são o nome, a idade e o overall.

A classe Treinador.java estende a classe Pessoa.java, e que herda os atributos e métodos da classe Pessoa e pode adicionar ou modificar comportamentos específicos do treinador. Tem dois métodos que nos dão a equipa que está a treinar.

A classe TreinadorAssistente.java é uma extensão da classe Treinador. Ela adiciona um atributo chamado posicaopreferida, que representa a posição preferida do treinador assistente. Além disso, a classe inclui métodos para acessarmos e modificarmos esse atributo getPosicaopreferida() e setPosicaopreferida() .Depois usamos o método toString()que dá-nos informações sobre o treinador assistente, incluindo nome, idade, equipa atual, posição preferida e overall. Este método utiliza getters para acessar os atributos protegidos da classe treinador getNome(),getIdade(), getEquipaAtreinar(), getOverall().

A classe TreinadorPrincipal.java estende-se à classe Treinador. Esta classe tem atributos que incluem listas para armazenar especializações e táticas preferidas, bem como um inteiro para representar o nível de moral. A classe possui métodos para acessar e manipular esses atributos, como adicionar especializações e táticas preferidas às listas. O construtor da classe permite-nos a criação de instâncias do treinador principal com informações iniciais, como nome, idade e overall. Através de métodos específicos, é possível obtermos e definirmos o nível de moral, acessar as especializações e táticas preferidas, e adicionarmos novos elementos a essas listas. Depois usamos um método toString que dá-nos as informações como nome, idade, equipa atual,especializações, táticas preferidas e overall.

Na classe SubMenu.java que estende-se à classe Menu.java cria um submenu com opções de registar um jogador , registar um treinador principal , registar um treinador assistente , registar uma equipa , registar um árbitro e voltar ao Menu\n" . Usamos métodos que verificam validam cada uma dessas opções e depois fazem o registo da opção que metemos no menu, os métodos que usamos percorrem as listas de ligas, equipas , treinadores principais , jogadores , treinadores assistentes verificam se as táticas são válidas ou não , as táticas de cada treinador , agressividade de cada jogador e todas as outras verificações de cada jogador , treinador principal assistente e táticas para depois exibir a informação pretendida.

Na classe PooProjetoFase2.java declaramos e inicializamos várias listas para armazenar instâncias de diferentes classes, como Equipa, TreinadorPrincipal, TreinadorAssistente, Jogador, Arbitro e Liga. Criamos instâncias de Equipa representando clubes de futebol, e são atribuídos treinadores principais a algumas equipas. As ligas usadas foram a Portuguesa , Espanhola e Alemã . São adicionados jogadores às equipas, com detalhes como nome, idade, posição e habilidades. Utilizamos a classe Leitura.java para ler informações de equipas a partir dos arquivos especificados.

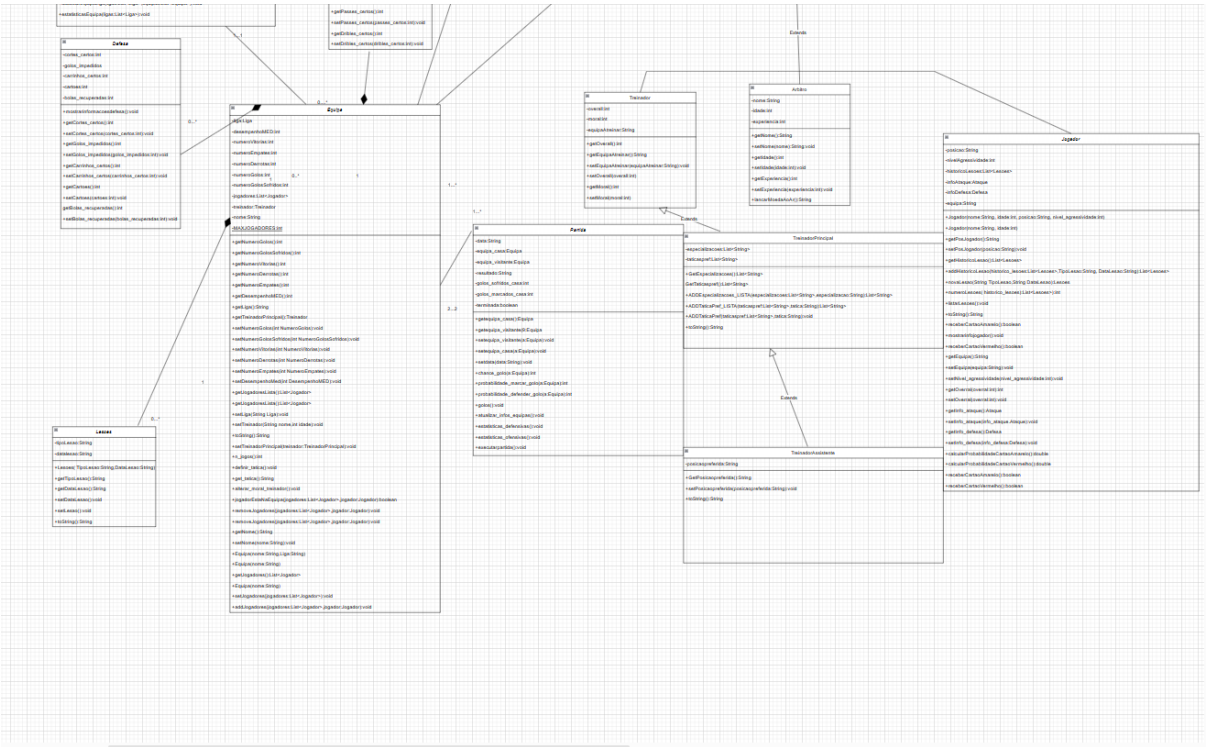
Conclusão

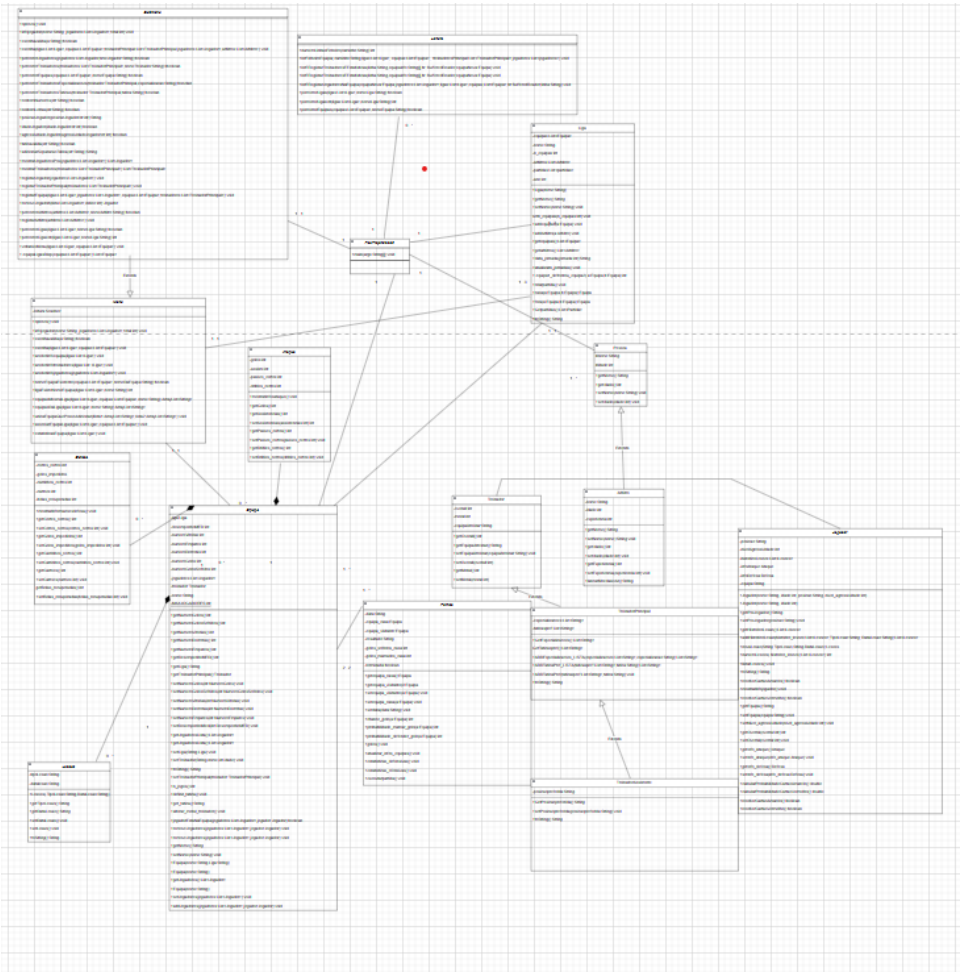
Este projeto visa criar uma simulação do jogo football manager ,que é o simulador mais realista de futebol que podemos encontrar, em que temos de gerir uma equipa de futebol e conseguimos inserir dados e acessar informações sobre treinadores, jogadores, partidas efetuadas e também equipas. Este jogo permite acesso detalhado a dados individuais e globais, desde estatísticas de desempenho dos jogadores até informações sobre lesões, táticas e histórico de lesões. A gestão de uma equipa é simulada de forma abrangente considerando fatores como a escolha de jogadores, estratégias do treinador, influência dos árbitros e a associação de equipas a apenas uma liga específica.

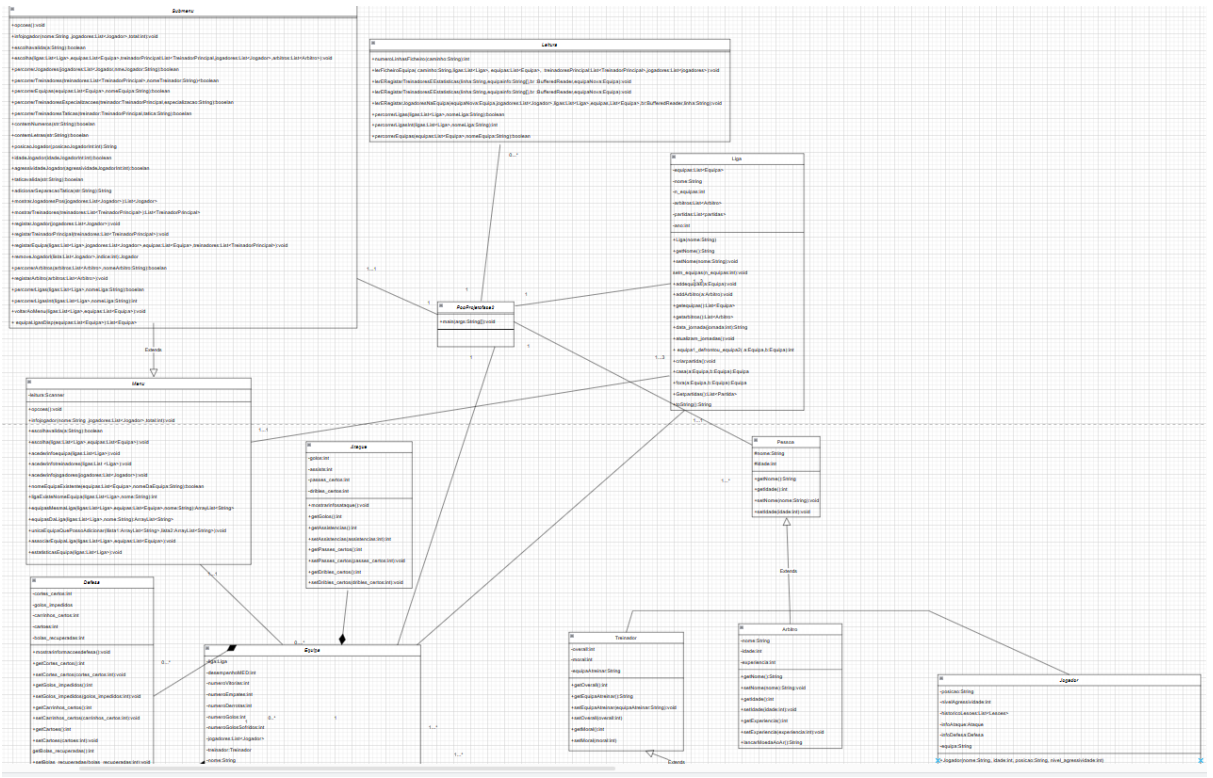
Este projeto não apenas oferece funcionalidades práticas para a gestão de uma equipa de futebol, mas também promove a reflexão sobre os possíveis desafios e limitações que podem surgir durante o processo.

Este projeto do jogo do fm não só atende às necessidades práticas de um utilizador fazer a gestão de equipas de futebol, mas também incorpora diversos fatores para proporcionar uma experiência completa e realista. Achamos que este projeto foi uma oportunidade muitíssimo boa de aplicar os conhecimentos de programação de forma prática e criativa, enquanto exploramos os diferentes aspectos envolvidos na gestão desportiva de um clube de futebol.

Diagrama UML







Código:

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.Random;

/**
 *
 * @author Renato
 */
public class Arbitro extends Pessoa {
    private int idade = 0;
    private int experiencia = 0;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public int getExperiencia() {
        return experiencia;
```

```
    }

    public void setExperiencia(int experiencia) {
        this.experiencia = experiencia;
    }

    // Método para lançar a moeda ao ar e determinar o resultado (cara
ou coroa)
    public String lancarMoedaAoAr(Equipa casa, Equipa fora) {
        Random lançamentoMoeda= new Random();
        int escolha_casa = lançamentoMoeda.nextInt(2); // Gera 0 ou 1
        int resultado = lançamentoMoeda.nextInt(2); // Gera 0 ou 1

        // Determina o resultado com base no valor gerado
        if (resultado == escolha_casa) {
            return casa.getNome();
        } else {
            return fora.getNome();
        }
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;

/**
 *
 * @author Renato
 */
public class Ataque {
```

```
private int golos = 0;
private int assistencias = 0;
private int passes_certos = 0;
private int dribles_certos = 0;

public int getGolos() {
    return golos;
}

public void setGolos(int golos) {
    this.golos = golos;
}

public int getAssistencias() {
    return assistencias;
}

public void setAssistencias(int assistencias) {
    this.assistencias = assistencias;
}

public int getPasses_certos() {
    return passes_certos;
}

public void setPasses_certos(int passes_certos) {
    this.passes_certos = passes_certos;
}

public int getDribles_certos() {
    return dribles_certos;
}

public void setDribles_certos(int dribles_certos) {
    this.dribles_certos = dribles_certos;
}
public void mostrarInfoAtaque() {
```

```
    System.out.println("Golos: " + golos);
    System.out.println("Assistências: " + assistencias);
    System.out.println("Passes certos: " + passes_certos);
    System.out.println("dribles_certos: " + dribles_certos);
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;

import java.util.List;

/**
 *
 * @author Paulo
 */
public interface Comum{
    public void opcoes();
    public boolean escolhavalida(String a);
    public boolean contemLetras(String str);
    public boolean contemNumeros(String str);

}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
```

```
package com.mycompany.poo.projetofase2;
import java.util.*;

/**
 *
 * @author Renato
 */
public class Defesa {
    private int cortes_certos = 0;
    private int golos_impedidos = 0;
    private int carrinhos_certos = 0;
    private int cartoesvermelhos = 0;
    private int cartoesamarelos = 0;
    private int bolas_recuperadas = 0;
    public void mostrarInfoDefesa(){
        System.out.println("Cortes certos: " + cortes_certos);
        System.out.println("Golos impedidos: " + golos_impedidos);
        System.out.println("Carrinhos certos: " + carrinhos_certos);
        System.out.println("Cartões Vermelhos: " + cartoesvermelhos);
        System.out.println("Cartões Amarelos: " + cartoesamarelos);
        System.out.println("Bolas recuperadas: " + bolas_recuperadas);
    }

    public int getCortes_certos() {
        return cortes_certos;
    }

    public void setCortes_certos(int cortes_certos) {
        this.cortes_certos = cortes_certos;
    }

    public int getGolos_impedidos() {
        return golos_impedidos;
    }

    public void setGolos_impedidos(int golos_impedidos) {
        this.golos_impedidos = golos_impedidos;
```

```
    }

    public int getCarrinhos_certos() {
        return carrinhos_certos;
    }

    public void setCarrinhos_certos(int carrinhos_certos) {
        this.carrinhos_certos = carrinhos_certos;
    }

    public int getCartoesvermelhos() {
        return cartoesvermelhos;
    }

    public void setCartoesvermelhos(int cartoes) {
        this.cartoesvermelhos = cartoes;
    }

    public int getCartoesamarelos() {
        return cartoesamarelos;
    }

    public void setCartoesamarelos(int cartoes) {
        this.cartoesamarelos = cartoes;
    }

    public int getBolas_recuperadas() {
        return bolas_recuperadas;
    }

    public void setBolas_recuperadas(int bolas_recuperadas) {
        this.bolas_recuperadas = bolas_recuperadas;
    }
}

/*
```

```
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Random;

/**
 *
 * @author Renato
 */
public class Equipa {
    private String nome = "";
    private String Liga="";
    private int desempenhoMedio=0;
    private int numeroVitorias=0;
    private int numeroEmpates=0;
    private int numeroDerrotas=0;
    private int numeroGolos=0;
    private int numeroGolosSofridos=0;
    private String tatica = "";
    private List<Jogador> jogadores=new ArrayList<Jogador>();
    private List<Jogador> jogadores_indisponiveis = new
ArrayList<Jogador>();
    private List <Jogador> jogadores_totais = new ArrayList <Jogador> ();
    private TreinadorPrincipal treinadorprincipal=new TreinadorPrincipal();
    private TreinadorAssistente treinadorassistente = new
TreinadorAssistente();
    //significa que a classe em si o numero maximo de jogadores é 11,dai
o uso de static
    //static ou seja pertence à classe e não à instancia
```

```
    private static final int MAX_JOGADORES = 11;

    public void setTatica(String tatica){

    this.tatica = tatica;
    }
    public Equipa(){

    }
    public Equipa(String nome,String Liga){
        this.nome=nome;
        this.Liga=Liga;
    }
    public Equipa(String nome){
        this.nome=nome;
    }
    public List<Jogador> getJogadores_indisponiveis(){
        return this.jogadores_indisponiveis;
    }

    public void setJogadores(List<Jogador> jogadores) {
        this.jogadores = jogadores;
    }

    public TreinadorAssistente getTreinadorAssistente(){
        return this.treinadorassistente;
    }
    public void addJogadores(List<Jogador> jogadores, Jogador jogador)
{
    if (this.jogadores.size() < MAX_JOGADORES) {
        if (!jogadorEstaNaEquipa(jogadores, jogador)) {
            jogador.setEquipa(this.getNome());
            this.jogadores.add(jogador);
        } else {
            System.out.println("Jogador já encontra-se na equipa");
        }
    } else {
```

```
        System.out.println("Equipa cheia");
    }
}

public boolean jogadorEstaNaEquipa(List<Jogador> jogadores, Jogador
jogador) {
    return this.jogadores.contains(jogador);
}
    public void removeJogadores(List<Jogador> jogadores,Jogador
jogador){
        for(int i=0;i<jogadores.size();i++){

if(jogadores.get(i).getNome().equalsIgnoreCase(jogador.getNome())){
                jogadores.remove(i);
                break;
            }
        }
    }
    public String getNome(){
        return this.nome;
    }
    public void setNome(String nome){
        this.nome = nome;
    }
    public int getNumeroGolos(){
        return numeroGolos;
    }

    public int getNumeroGolosSofridos(){
        return numeroGolosSofridos;
    }

    public int getNumeroVitorias(){
        return numeroVitorias;
    }

    public int getNumeroDerrotas(){
```

```
    return numeroDerrotas;
}

public int getNumeroEmpates(){
    return numeroEmpates;
}

public int getDesempenhoMED(){
    return desempenhoMedio;
}

public String getLiga(){
    return Liga;
}

public TreinadorPrincipal getTreinadorPrincipal(){
    return this.treinadorprincipal;
}

public void setNumeroGolos(int NumeroGolos) {
    this.numeroGolos = NumeroGolos;
}

public void setNumeroGolosSofridos(int NumeroGolosSofridos) {
    this.numeroGolosSofridos = NumeroGolosSofridos;
}

public void setNumeroVitorias(int NumeroVitorias) {
    this.numeroVitorias = NumeroVitorias;
}

public void setNumeroDerrotas(int NumeroDerrotas) {
    this.numeroDerrotas = NumeroDerrotas;
}

public void setNumeroEmpates(int NumeroEmpates) {
    this.numeroEmpates = NumeroEmpates;
```

```
    }

    public void setDesempenhoMed(int DesempenhoMED) {
        this.desempenhoMedio = DesempenhoMED;
    }

    public List<Jogador> getJogadoresLista(){
        return jogadores;
    }

    public void setLiga(String Liga){
        this.Liga=Liga;
    }

    public void setTreinador(String nome,int idade){
        this.treinadorprincipal.setNome(nome);
        this.treinadorprincipal.setIdade(idade);
    }
    public void setTreinadorPrincipal(TreinadorPrincipal treinador){
        this.treinadorprincipal=treinador;
        this.treinadorprincipal.setEquipaAtreinar(this.getNome());

    }

    public String toString(){
        return "Equipa: " + nome + "\nTreinador: " +
treinadorprincipal.getNome() + "\nLiga: " + Liga +
                "\nDesempenho médio: " + desempenhoMedio + "\nNumero
de vitorias: " + numeroVitorias
                +"\nNumero de derrotas: " + numeroDerrotas +
                "\nNumero de empates: " + numeroEmpates +
                "\nNumero de golos marcados: " + numeroGolos +
                "\nNumero de golos sofridos: " + numeroGolosSofridos;

    }

    public int n_jogos(){
```

```
        int n_jogos = this.numeroDerrotas + this.numeroEmpates +
this.numeroVitorias;
        return n_jogos;
    }



    public void definir_tatica(){
        int n_atacantes = 0;
        int n_medios = 0;
        int n_defesas = 0;
        for (int i = 0; i<this.jogadores_totais.size(); i++){

if(this.jogadores_totais.get(i).getPosJogador().equalsIgnoreCase("MEDI
O")){
                n_medios = n_medios + 1;
            }
            else
if(this.jogadores_totais.get(i).getPosJogador().equalsIgnoreCase("ATAC
ANTE")){
                n_atacantes = n_atacantes + 1;
            }
            else
if(this.jogadores_totais.get(i).getPosJogador().equalsIgnoreCase("DEFE
SA")){
                n_defesas = n_defesas + 1;
            }
        }
        if(n_medios + n_atacantes + n_defesas == 10){
          this.tatica = n_defesas + "-" + n_medios + "-" + n_atacantes;
        }
        else{
            System.out.println("Numero de jogadores invalido");
        }
    }

    public String get_tatica(){
```

```
        return this.tatica;
    }

    public void alterar_moral_treinador(){
        boolean tatica_preferida = false;
        for(int i = 0; i < this.treinadorprincipal.GetTaticaspref().size() ; i++){

if(this.treinadorprincipal.GetTaticaspref().get(i).equals(this.tatica)){
                tatica_preferida = true;
                break;
            }
        }
        int overall = this.treinadorprincipal.getOverall();
        if(!tatica_preferida){
            if(overall > 10){
                this.treinadorprincipal.setMoral(overall - 10);
            }
            else{
                this.treinadorprincipal.setMoral(1);
            }
        }
        else{
            this.treinadorprincipal.setMoral(overall);
        }
    }

    public void resetarequipa(){
        this.desempenhoMedio = 0;
        this.numeroDerrotas = 0;
        this.numeroEmpates = 0;
        this.numeroGolos = 0;
        this.numeroGolosSofridos = 0;
        this.numeroVitorias = 0;
    }

    public void verificarindisponiveis(){
        for(int j = 0; j < this.jogadores_indisponiveis.size(); j++){
```

```
            Jogador atual = jogadores_indisponiveis.get(j);
            if(atual.getdias_indisponiveis() <= 0){
                this.jogadores.add(atual);
                this.jogadores_indisponiveis.remove(atual);
            }
            else{
                atual.setdias_indisponiveis(atual.getdias_indisponiveis() - 1);
            }
        }
    }

    public void setTreinadorAssistente(TreinadorAssistente treinador){
        this.treinadorassistente=treinador;
        this.treinadorassistente.setEquipaAtreinar(this.getNome());
    }

    public void treinarjogadores(){
        Random aleatorio = new Random();
        int chance_melhoria = this.treinadorassistente.getOverall();
        for(int i = 0; i < this.jogadores.size(); i++){

if(this.treinadorassistente.getPosicaopreferida().equals(this.jogadores.ge
t(i).getPosJogador())){
                if(chance_melhoria < 95){
                    chance_melhoria = chance_melhoria + 5;
                }
                else{
                    chance_melhoria = 99;
                }
                if((aleatorio.nextInt(2001 - 1) + 1) <= chance_melhoria){
                    if(this.jogadores.get(i).getOverall() < 99){

this.jogadores.get(i).setOverall(this.jogadores.get(i).getOverall() + 1);
                    }
                    else{
                        this.jogadores.get(i).setOverall(99);
                    }
```

```
            }
        }
    }
    public void definir_jogadores_totais(){
        List <Jogador> jogadores_totais_temp = new ArrayList();
        for(int i = 0; i < this.jogadores.size(); i++){
            jogadores_totais_temp.add(this.jogadores.get(i));
        }
        for(int j = 0; j < this.jogadores_indisponiveis.size(); j++){
            jogadores_totais_temp.add(this.jogadores_indisponiveis.get(j));
        }
        this.jogadores_totais = jogadores_totais_temp;
    }

    public List<Jogador> getJogadoresatuais(){
        return this.jogadores_totais;
    }

    public int pontos(){
        int pontos = 0;
        pontos = (this.numeroVitorias * 3) + (this.numeroEmpates);
        return pontos;
    }

    public void DesempenhoMedio(){
        if(this.n_jogos() != 0){
        this.desempenhoMedio = this.pontos()/this.n_jogos();
        }
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;

/**
 *
 * @author Paulo
 */
public class Jogador extends Pessoa {
    private String posicao="";
    private String equipa="";
    private List<Lesoes> historico_lesoes=new ArrayList<>();
    private Ataque info_ataque = new Ataque();
    private Defesa info_defesa = new Defesa();
    private int nivel_agressividade=0;
    private int dias_indisponiveis = 0;

    public int getdias_indisponiveis(){
        return dias_indisponiveis;
    }
    public void setdias_indisponiveis(int dias){
        this.dias_indisponiveis = dias;
    }

    public Jogador(String nome,int idade,String posicao,int
nivel_agressividade){
        this.nome = nome;
        this.idade = idade;
        this.posicao=posicao;
        this.nivel_agressividade=nivel_agressividade;
    }
    public Jogador(String nome,int idade,String posicao,int
nivel_agressividade, int overall){
        this.nome = nome;
        this.idade = idade;
```

```
    this.posicao=posicao;
    this.nivel_agressividade=nivel_agressividade;
    this.overall = overall;
}
public Jogador(){

}
public Jogador(String nome,int idade){
    this.nome=nome;
    this.idade=idade;
}
public String getEquipa() {
    return equipa;
}

public void setEquipa(String equipa) {
    this.equipa = equipa;
}
public void setNivel_agressividade(int nivel_agressividade){
    this.nivel_agressividade=nivel_agressividade;
}


public String getPosJogador(){
    return posicao;
}

public void setPosJogador(String posicao){
    this.posicao=posicao;
}

public List<Lesoes> getHistoricoLesao(){
    return historico_lesoes;
}

/*Adiciona uma nova lesao e retorna todas as lesoes do jogador*/
```

```
    public List<Lesoes> adicionarHistoricoLesao(List<Lesoes>
historico_lesoes,String TipoLesao,String DataLesao){
        historico_lesoes.add(novaLesao(TipoLesao,DataLesao));
        return historico_lesoes;
    }
    public void novaLesaoaleatoria(String DataLesao){
        Random aleatorio = new Random();
        List <String> tipos_lesoes = new ArrayList();
        tipos_lesoes.add("Lesão no joelho");
        tipos_lesoes.add("Lesão no tornozelo");
        tipos_lesoes.add("Lesão na coxa");
        tipos_lesoes.add("Fratura");
        tipos_lesoes.add("Lesão no braço");
        tipos_lesoes.add("Lesão na cabeça");
        tipos_lesoes.add("Lesão no pescoço");
        tipos_lesoes.add("Lesão no tronco");
        int lesao_aleatoria = aleatorio.nextInt(tipos_lesoes.size() - 0) + 0;
        this.historico_lesoes.add(novaLesao(DataLesao,
tipos_lesoes.get(lesao_aleatoria)));
    }
    public Lesoes novaLesao(String TipoLesao,String DataLesao){
        Lesoes temp_lesao= new Lesoes(TipoLesao,DataLesao);
        return temp_lesao;
    }

    public int numeroLesoes(List<Lesoes> historico_lesoes){
        return historico_lesoes.size();
    }

    public Ataque getInfo_ataque() {
        return info_ataque;
    }

    public void setInfo_ataque(Ataque info_ataque) {
        this.info_ataque = info_ataque;
    }
```

```
    public Defesa getInfo_defesa() {
        return info_defesa;
    }

    public void setInfo_defesa(Defesa info_defesa) {
        this.info_defesa = info_defesa;
    }

    public void listarLesoes(){
        String temp = new String();
        for(int i=0;i<historico_lesoes.size();i++){
            System.out.println(historico_lesoes.toString());
        }
    }
    public String toString(){
        return "Nome Jogador: " + nome + "\nIdade: " + idade +"\nPosicao:
" + posicao +
                "\nHistorico de Lesoes: " + historico_lesoes +
                "\nNivel Agressividade: " + this.nivel_agressividade;

    }

    public int getagressividade(){
        return this.nivel_agressividade;
    }

//    // Método para determinar a probabilidade de receber um cartão
amarelo
//   private double calcularProbabilidadeCartaoAmarelo() {
//        double baseProbabilidadeCartaoAmarelo = 0.15; // Probabilidade
base
//        double incrementoPorAgressividade = 0.05; // Incremento por
unidade de agressividade
//
//        return baseProbabilidadeCartaoAmarelo + (nivel_agressividade *
incrementoPorAgressividade);
//    }
```

```
//       // Método para determinar a probabilidade de receber um cartão
vermelho
//    private double calcularProbabilidadeCartaoVermelho() {
//        double baseProbabilidadeCartaoVermelho = 0.01; // Probabilidade
base
//        double incrementoPorAgressividade = 0.04; // Incremento por
unidade de agressividade
//
//        return baseProbabilidadeCartaoVermelho + (nivel_agressividade *
incrementoPorAgressividade);
//    }
//     // Método para verificar se o jogador recebe um cartão amarelo
//    public boolean receberCartaoAmarelo() {
//        Random random = new Random();
//        double probabilidade = calcularProbabilidadeCartaoAmarelo();
//
//        // Gera um número aleatório entre 0 e 1
//        double aleatorio = random.nextDouble();
//
//        // Verifica se o jogador recebe um cartão amarelo com base na
probabilidade
//        return aleatorio < probabilidade;//Se o valor gerado for menor do
que a probabilidade retorna true
//        // indicando que o jogador deve de receber cartão amarelo,caso
contrário false indicando que não deve de receberer cartão amarelo
//    }
//
//    // Método para verificar se o jogador recebe um cartão vermelho
//    public boolean receberCartaoVermelho() {
//        Random random = new Random();
//        double probabilidade = calcularProbabilidadeCartaoVermelho();
//
//        // Gera um número aleatório entre 0 e 1
//        double aleatorio = random.nextDouble();
//
//        // Verifica se o jogador recebe um cartão vermelho com base na
probabilidade
```

```
//        return aleatorio < probabilidade;//Se o valor gerado for menor do
que a probabilidade retorna true
//        // indicando que o jogador deve de receber cartão vermelho,caso
contrário false indicando que não deve de receberer cartão vermelho
//    }

    public void mostrarinfojogador(){
        System.out.println("Nome: " + this.nome);
        System.out.println("Idade: " + this.idade);
        System.out.println("posição: " + this.posicao);
        System.out.println("nivel de agressividade: " +
this.nivel_agressividade);
        System.out.println("Dias indisponível: " + this.dias_indisponiveis);
        System.out.println("Historico de lesoes\n");
        for(int i = 0; i< this.historico_lesoes.size(); i++){
            System.out.println("Data de lesão:
"+this.historico_lesoes.get(i).getDataLesao()+"\nTipo de Lesão: "
            + this.historico_lesoes.get(i).getTipoLesao()+ ".\n\n");
        }
        System.out.println("\nEstatísticas de ataque");
        this.info_ataque.mostrarInfoAtaque();
        System.out.println("\nEstatísticas de defesa");
        this.info_defesa.mostrarInfoDefesa();
        System.out.println();
    }
}

package com.mycompany.poo.projetofase2;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.nio.charset.Charset;
import java.util.List;

public class Leitura{
```

```
    public int numeroLinhasFicheiro(String caminho) throws IOException{
    FileReader inStream = new FileReader(caminho);
    BufferedReader bR = new BufferedReader(inStream);
    String linha=bR.readLine();
    int numeroLinhas=0;
    while(linha!=null){
        numeroLinhas++;
        linha=bR.readLine();
    }
    return numeroLinhas;
}
    public void lerFicheiroEquipa(String caminho,List<Liga> ligas,
List<Equipa> equipas,
    List<TreinadorPrincipal> treinadoresPrincipal,List<Jogador>
jogadores,
    List<Arbitro> arbitros,List<TreinadorAssistente>
treinadoresAssistente)throws IOException{
        int tamanhoFicheiro= numeroLinhasFicheiro(caminho);
        FileReader inStream = new FileReader(caminho);
        BufferedReader bR = new BufferedReader(inStream);
        Equipa equipaNova = new Equipa();
        TreinadorPrincipal treinadorPrincipalNovo = new
TreinadorPrincipal();
        TreinadorAssistente treinadorAssistenteNovo = new
TreinadorAssistente();



        String linha = bR.readLine();
        String[] equipaInfo={};


            lerERegistarTreinadoresEEstatisticas(linha, equipaInfo, bR,
equipaNova,
            treinadorPrincipalNovo, treinadorAssistenteNovo,
treinadoresAssistente, treinadoresPrincipal);

lerERegistarJogadoresNaEquipa(equipaNova,jogadores,ligas,equipas,b
R,linha);
```

```
    }
    public void lerERegistarTreinadoresEEstatisticas(String linha,String[]
equipaInfo,BufferedReader bR,Equipa equipaNova
            ,TreinadorPrincipal treinadorPrincipalNovo,TreinadorAssistente
treinadorAssistenteNovo
            ,List<TreinadorAssistente>
treinadoresAssistente,List<TreinadorPrincipal> treinadoresPrincipal)
throws IOException{
        linha = bR.readLine();
        equipaInfo=linha.split("\\|");

        equipaNova.setNome(equipaInfo[0]);
            equipaNova.setLiga(equipaInfo[1]);
            int desempenhoMed=Integer.parseInt(equipaInfo[2]);
            equipaNova.setDesempenhoMed(desempenhoMed);

            int numeroVitorias=Integer.parseInt(equipaInfo[3]);
            equipaNova.setNumeroVitorias(numeroVitorias);

            int numeroEmpates=Integer.parseInt(equipaInfo[4]);
            equipaNova.setNumeroEmpates(numeroEmpates);

            int umeroDerrotas=Integer.parseInt(equipaInfo[5]);
            equipaNova.setNumeroDerrotas(umeroDerrotas);

            int numeroGolos=Integer.parseInt(equipaInfo[6]);
            equipaNova.setNumeroGolos(numeroGolos);

            int numeroGolosSofridos=Integer.parseInt(equipaInfo[7]);
            equipaNova.setNumeroGolosSofridos(numeroGolosSofridos);

            equipaNova.setTatica(equipaInfo[8]);

            treinadorPrincipalNovo.setNome(equipaInfo[9]);
```

```
            int idadeTreinadorPrincipal=Integer.parseInt(equipaInfo[10]);
            treinadorPrincipalNovo.setIdade(idadeTreinadorPrincipal);

            String[] treinadorPrincipalEsp=equipaInfo[11].split("\\#");
            for(int i=0;i<treinadorPrincipalEsp.length;i++){

treinadorPrincipalNovo.ADDEspecializacoes(treinadorPrincipalNovo.Get
Especializacoes(), treinadorPrincipalEsp[i]);
            }

            String[] treinadorPrincipalTaticasPref=equipaInfo[12].split("\\/");
            for(int i=0;i<treinadorPrincipalTaticasPref.length;i++){

treinadorPrincipalNovo.ADDTaticaPref(treinadorPrincipalNovo.GetTatica
spref(), treinadorPrincipalTaticasPref[i]);
            }

            int overallTreinadorPrincipal=Integer.parseInt(equipaInfo[13]);
            treinadorPrincipalNovo.setOverall(overallTreinadorPrincipal);

            treinadorAssistenteNovo.setNome(equipaInfo[14]);

            int idadeTreinadorAssistente=Integer.parseInt(equipaInfo[15]);
            treinadorAssistenteNovo.setIdade(idadeTreinadorAssistente);

            treinadorAssistenteNovo.setPosicaopreferida(equipaInfo[16]);

            int overallTreinadorAssistente=Integer.parseInt(equipaInfo[17]);
            treinadorAssistenteNovo.setOverall(overallTreinadorAssistente);

            treinadoresPrincipal.add(treinadorPrincipalNovo);
            treinadoresAssistente.add(treinadorAssistenteNovo);

            int
indiceTreinadorPrincipal=treinadoresPrincipal.indexOf(treinadorPrincipal
Novo);
```

```
            int
indiceTreinadorAssistente=treinadoresAssistente.indexOf(treinadorAssis
tenteNovo);

            if(indiceTreinadorPrincipal!=-1){

equipaNova.setTreinadorPrincipal(treinadoresPrincipal.get(indiceTreinad
orPrincipal));
            }
            if(indiceTreinadorAssistente!=-1){

equipaNova.setTreinadorAssistente(treinadoresAssistente.get(indiceTrei
nadorAssistente));
            }
//            System.out.print(equipaNova);
//
System.out.print(treinadoresAssistente.get(indiceTreinadorAssistente));
//
System.out.print(treinadoresPrincipal.get(indiceTreinadorPrincipal));

    }
    public void lerERegistarJogadoresNaEquipa(Equipa
equipaNova,List<Jogador> jogadores,List<Liga> ligas,List<Equipa>
equipas,BufferedReader bR,String linha) throws IOException{
        for(int i=0;i<11;i++){
                linha = bR.readLine();
                String[] equipaJogadores=linha.split("\\|");
                Jogador jogadorNovo = new Jogador();
                jogadorNovo.setNome(equipaJogadores[0]);

                int idadeJogadorNovo=Integer.parseInt(equipaJogadores[1]);
                jogadorNovo.setIdade(idadeJogadorNovo);
                jogadorNovo.setPosJogador(equipaJogadores[2]);

                int
agressividadeJogadorNovo=Integer.parseInt(equipaJogadores[3]);
```

```
jogadorNovo.setNivel_agressividade(agressividadeJogadorNovo);

            int overallJogadorNovo=Integer.parseInt(equipaJogadores[4]);
            jogadorNovo.setOverall(overallJogadorNovo);

            jogadores.add(jogadorNovo);
            int indiceJogador=jogadores.indexOf(jogadorNovo);

            if(indiceJogador!=-1){
               equipaNova.addJogadores(jogadores,
jogadores.get(indiceJogador));
            }

         }
         equipas.add(equipaNova);

      int equipaIntPos=-1;
      for(int i=0;i<equipas.size();i++){

if(equipas.get(i).getNome().equalsIgnoreCase(equipaNova.getNome())){
            equipaIntPos=i;
         }
      }

      int valorLiga=-1;
      //se a liga existir
      if(percorrerLigas(ligas,equipas.get(equipaIntPos).getLiga())){

valorLiga=percorrerLigasInt(ligas,equipas.get(equipaIntPos).getLiga());
         ligas.get(valorLiga).addequipa(equipas.get(equipaIntPos));
      }
      else{
         Liga ligaNova= new Liga(equipas.get(equipaIntPos).getLiga());
         ligaNova.addequipa(equipas.get(equipaIntPos));
         ligas.add(ligaNova);
      }
```

```
    }
    public boolean percorrerLigas(List<Liga> ligas,String nomeLiga){
        for(int i=0;i<ligas.size();i++){
            if(ligas.get(i).getNome().equalsIgnoreCase(nomeLiga)){
                return true;
            }
        }
        return false;
    }
    public int percorrerLigasInt(List<Liga> ligas,String nomeLiga){
        for(int i=0;i<ligas.size();i++){
            if(ligas.get(i).getNome().equalsIgnoreCase(nomeLiga)){
                return i;
            }
        }
        return -1;
    }
    public boolean percorrerEquipas(List<Equipa> equipas,String
nomeEquipa){
        for(int i=0;i<equipas.size();i++){
            if(nomeEquipa.equalsIgnoreCase(equipas.get(i).getNome())){
                return true;
            }
        }
        return false;
    }
    public void lerFicheiroArbitro(String caminho,List<Liga> ligas,
List<Equipa> equipas,
    List<TreinadorPrincipal> treinadoresPrincipal,List<Jogador>
jogadores,
    List<Arbitro> arbitros,List<TreinadorAssistente>
treinadoresAssistente)throws IOException{
    FileReader inStream = new FileReader(caminho);
    BufferedReader bR = new BufferedReader(inStream);
    //NOME|IDADE|EXPERIENCIA

    //cada linha 1 arbitro
```

```
    int tamanhoFicheiro= numeroLinhasFicheiro(caminho);
    String linha = bR.readLine();
    String[] arbitroInfo={};

    int ultimo=0;
    for(int i=0;i<tamanhoFicheiro-1;i++){
    linha = bR.readLine();
    arbitroInfo=linha.split("\\|");

    Arbitro arbitroNovo = new Arbitro();
    arbitroNovo.setNome(arbitroInfo[0]);//3-6

    int idadeArbitroNovo=Integer.parseInt(arbitroInfo[1]);//4-7
    arbitroNovo.setIdade(idadeArbitroNovo);

    int experienciaArbitroNovo=Integer.parseInt(arbitroInfo[2]);//5-8

    arbitroNovo.setExperiencia(experienciaArbitroNovo);
    arbitros.add(arbitroNovo);
//    System.out.println(arbitroNovo);
    }

    }
}
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;

/**
 *
 * @author Paulo
```

```
 */
public class Lesoes {
    private String tipoLesao="";
    private String dataLesao="";

    public Lesoes(String TipoLesao,String DataLesao){
        this.tipoLesao=TipoLesao;
        this.dataLesao=DataLesao;
    }

    public String getTipoLesao(){
        return tipoLesao;
    }

    public void setLesao(String TipoLesao){
        this.tipoLesao=TipoLesao;
    }

    public String getDataLesao(){
        return dataLesao;
    }

    public void setDataLesao(String DataLesao){
        this.dataLesao=DataLesao;
    }

    @Override
    public String toString(){
        return "\nTipo de Lesao: " + tipoLesao +",Data Lesao: "+dataLesao;
    }

}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;
import java.lang.Math.*;

/**
 *
 * @author Renato
 */
public class Liga {
    private String nome = "";
    private int jornada = 1;

    public int getjornada(){
        return this.jornada;
    }
    private List <Equipa> equipas = new ArrayList();
    private List <Arbitro> arbitros = new ArrayList();
    private List <Partida> partidas = new ArrayList();
    private int ano = 2023;


    public Liga(String nome){
          this.nome = nome;
    }
    public Liga(){

    }
    public String getNome(){
        return this.nome;
    }
    public void setNome(String nome){
        this.nome = nome;
    }
```

```
public void addequipa(Equipa a){
    this.equipas.add(a);
}
public void addarbitro(Arbitro a){
    this.arbitros.add(a);
}

public List<Equipa> getequipas(){
    return this.equipas;
}
public List<Arbitro> getarbitros(){
    return this.arbitros;
}

public void atualizarn_jornadas(){
    if(equipas.size() > 1){
    int n_jogos = equipas.get(1).n_jogos();
    boolean atualizar = true;
    for(int i=0; i< equipas.size(); i++){
        if(equipas.get(i).n_jogos() != n_jogos){
            atualizar = false;
            break;
        }
    }
    if(atualizar){
        if((n_jogos + 1) <= this.equipas.size()){
        this.jornada = n_jogos +1;
        for(int j = 0; j < this.equipas.size(); j++){
            this.equipas.get(j).treinarjogadores();
            this.equipas.get(j).verificarindisponiveis();
        }
        }
        else{
            System.out.println("Começando nova epoca...");
            comecarnovaepoca();
        }
    }
```

```
        }
    }

    public void comecarnovaepoca(){
        this.jornada = 1;
        this.partidas = new ArrayList();
        this.ano = this.ano + 1;
        for(int i = 0; i < this.equipas.size(); i ++){
            this.equipas.get(i).resetarequipa();
        }
        aumentaridades();
    }

    public void aumentaridades(){
        for (int i = 0; i < this.arbitros.size(); i++){
            arbitros.get(i).setIdade(arbitros.get(i).getIdade() + 1);
        }
        for(int j = 0; j<this.equipas.size(); j++){
            Equipa equipa_atual = this.equipas.get(j);
            for(int k = 0; k < equipa_atual.getJogadoresLista().size(); k++){
                Jogador atual = equipa_atual.getJogadoresLista().get(k);
                atual.setIdade(atual.getIdade() + 1);
            }

equipa_atual.getTreinadorPrincipal().setIdade(equipa_atual.getTreinador
Principal().getIdade() + 1);

equipa_atual.getTreinadorAssistente().setIdade(equipa_atual.getTreinad
orAssistente().getIdade() + 1);
        }
    }

    public int equipa1_defrontou_equipa2(Equipa a, Equipa b){
        int n_vezes_defrontou = 0;
        for(int i=0; i < partidas.size(); i++){
            Partida partida = partidas.get(i);
```

```
            if((partida.getequipa_casa() == a &&
partida.getequipa_visitante() == b)||(partida.getequipa_casa() == b &&
partida.getequipa_visitante() == a)){
                n_vezes_defrontou = n_vezes_defrontou + 1;
            }
        }
        return n_vezes_defrontou;
    }

    public Equipa casa(Equipa a, Equipa b){ //função usada para saber
qual equipa jogou em casa no primeiro jogo(só é útil para equipas que
so jogaram entre si uma vez esta temporada)
        Equipa casa = new Equipa();
        for(int i = 0; i < partidas.size(); i++){
            Partida partida = partidas.get(i);
            if((partida.getequipa_casa() == a &&
partida.getequipa_visitante() == b)){
                casa = a;
            }
            else if((partida.getequipa_casa() == b &&
partida.getequipa_visitante() == a)){
                casa = b;
            }
            else{
                casa = null;
            }
        }
        return casa;
    }
        public Equipa fora(Equipa a, Equipa b){ //função usada para saber
qual equipa jogou fora no primeiro jogo(só é útil para equipas que so
jogaram entre si uma vez esta temporada)
        Equipa fora = new Equipa();
        for(int i = 0; i < partidas.size(); i++){
            Partida partida = partidas.get(i);
            if((partida.getequipa_casa() == a &&
partida.getequipa_visitante() == b)){
```

```
                fora = b;
            }
            else if((partida.getequipa_casa() == b &&
partida.getequipa_visitante() == a)){
                fora = a;
            }
            else{
                fora = null;
            }
        }
        return fora;
    }


    public String data_jornada(int jornada){
        //temporada começa 11 de agosto(+ou-)
        int dia = 0;
        int mes = 0;
        int ano_d = 0;
        int dias_totais = jornada * 10 ;//uma jornada ocorre a cada x dias;
        if(dias_totais >= 144){
            int dias_totais_temp = dias_totais - 143;
            mes = 1 + (dias_totais_temp/30);
            dia = (dias_totais_temp%30)*30;
            ano_d = this.ano + 1;
        }
        else{
            int dias_totais_temp = 11 + dias_totais;
            mes = 8 + (dias_totais_temp/30);
            dia = (dias_totais_temp%30);
            if(dia == 0){
                dia = 31;
            }
            ano_d = this.ano;
        }
        String data = dia + "-" + mes + "-" + ano_d;
        return data;
```

```
    }

    public void criarpartida(){
        boolean continuar = false;
        Random aleatorio = new Random();
        Partida nova = new Partida();
        if(this.arbitros.size() > 0)
        {
        int ind_arbitro = aleatorio.nextInt(this.arbitros.size());
        nova.setarbitro(this.arbitros.get(ind_arbitro));
        if(this.jornada < this.equipas.size()){
            while(!continuar){
                int equipa1 = aleatorio.nextInt((equipas.size()) - 0) + 0;
                int equipa2 = aleatorio.nextInt((equipas.size()) - 0) + 0;
                if(equipa1 != equipa2 &&
(((equipa1_defrontou_equipa2(equipas.get(equipa1),equipas.get(equipa
2))))==0)){
                    nova.setequipa_casa(equipas.get(equipa1));
                    nova.setequipa_visitante(equipas.get(equipa2));
                    continuar = true;
                }
                else{
                    continuar = false;
                }
            }
        }
        else if(this.jornada >=this.equipas.size() && this.jornada <=
this.equipas.size() * 2 - 2){
            while(!continuar){
            int equipa1 = aleatorio.nextInt(equipas.size()) + 0;
            int equipa2 = aleatorio.nextInt(equipas.size()) + 0;
            if(equipa1 != equipa2 &&
((equipa1_defrontou_equipa2(equipas.get(equipa1),equipas.get(equipa2
))))==1){

nova.setequipa_visitante(casa(equipas.get(equipa1),equipas.get(equipa
2)));
```

```
nova.setequipa_casa(fora(equipas.get(equipa1),equipas.get(equipa2)));
                continuar = true;
            }
            else{
                continuar = false;
            }
            }

        }
        nova.setdata(data_jornada(this.jornada));
        this.partidas.add(nova);
    }
    else{
        System.out.println("Não há arbitros!");
    }
    }

    public List <Partida> Getpartidas(){
        return this.partidas;
    }

    public String toString(){
        return "Liga: " + nome + "\nJornada: " + jornada + "\nEquipas: " +
equipas;
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.io.IOException;
```

```
import java.util.*;

/**
 *
 * @author Paulo
 */
public class Menu implements Comum {
    Scanner leitura = new Scanner(System.in);
    public void opcoes(){
        System.out.println("1. Aceder informações de um jogador \n2.
Aceder informações de um treinador \n"
                + "3. Aceder informações de uma equipa \n4. Criar uma
partida \n"
                + "5. Estatísticas de Equipa" +"\n6. Mostrar tabela de uma
liga" + "\n7. Ir para o submenu"+"\n8. Sair");
    }
    public boolean escolhavalida(String a){
        if(a.isEmpty()){
            return false;
        }
        else if(a.length() > 1){
            return false;
        }
        else if(a.length() == 1){
            int b = a.charAt(0); //converte char para o valir int da tabela
ASCII
            if ((b >=49) && (b <= 57)){
                return true;
            }
            else{
                return false;
            }
        }
        else{
            return false;
        }
    }
```

```
    public void acederinfojogadores(List <Liga> ligas){
        System.out.println("Digite o nome completo do jogador: ");
        String nome = leitura.nextLine();
        boolean existe = false;
        for(int j=0; j<ligas.size(); j++){
            for(int i=0; i< ligas.get(j).getequipas().size(); i++){
                for(int k = 0;
k<ligas.get(j).getequipas().get(i).getJogadoresatuais().size(); k++){

if(nome.equalsIgnoreCase(ligas.get(j).getequipas().get(i).getJogadoresat
uais().get(k).getNome())){
                        System.out.println("Equipa: " +
ligas.get(j).getequipas().get(i).getNome());

ligas.get(j).getequipas().get(i).getJogadoresatuais().get(k).mostrarinfojog
ador();
                        existe = true;
                        break;
                    }
                    else{
                        continue;
                    }
                    }

                }
            }
        if(!existe){
            System.out.println("Nome não existe!");
        }
    }

    public void acederinfotreinadores(List <Liga> ligas){
        System.out.println("Digite o nome completo do Treinador: ");
        String nome = leitura.nextLine();
        boolean existe = false;
```

```
        for(int j=0; j<ligas.size(); j++){
            for(int i=0; i< ligas.get(j).getequipas().size(); i++){

if(nome.equalsIgnoreCase(ligas.get(j).getequipas().get(i).getTreinadorPri
ncipal().nome)){

System.out.println(ligas.get(j).getequipas().get(i).getTreinadorPrincipal())
;
                    existe = true;
                    break;
                }
                else{
                    continue;
                }
            }
        }
        if(!existe){
            System.out.println("Nome não existe!");
        }
    }

    public void acederinfoequipa(List <Liga> ligas){
        System.out.println("Digite o nome da equipa: ");
        String nome = leitura.nextLine();
        boolean existe = false;
        for(int j=0; j<ligas.size(); j++){
            for(int i=0; i< ligas.get(j).getequipas().size(); i++){

if(nome.equalsIgnoreCase(ligas.get(j).getequipas().get(i).getNome())){
                    ligas.get(j).getequipas().get(i).DesempenhoMedio();
                    System.out.println(ligas.get(j).getequipas().get(i));
                    System.out.println("Jogadores:");
                    for(int k=0;
k<ligas.get(j).getequipas().get(i).getJogadoresatuais().size();k++){

ligas.get(j).getequipas().get(i).getJogadoresatuais().get(k).mostrarinfojog
ador();
```

```
                    }
                    existe = true;
                    break;
                }
                else{
                    continue;
                }
            }
        }
        if(!existe){
            System.out.println("Nome não existe!");
        }
    }

    public void criarpartida(List <Liga> ligas){
        boolean continuar = false;
        Liga escolhida = new Liga();
        while(!continuar ){
            System.out.println("Digite a liga onde quer criar uma partida: ");
            String escolha = leitura.nextLine();
            for(int i = 0; i < ligas.size(); i ++){
                if(ligas.get(i).getNome().equalsIgnoreCase(escolha)){
                    continuar = true;
                    escolhida = ligas.get(i);
                    break;
                }
            }
            if(!continuar){
                System.out.println("Liga não existe");
            }
            else{
                System.out.println("Criando uma partida da liga escolhida...");
            }
        }
        if(escolhida.getequipas().size() >= 2 &&
escolhida.getequipas().size()%2 ==0 && escolhida.getarbitros().size() >
0){
```

```
        escolhida.atualizarn_jornadas();
        escolhida.criarpartida();
        System.out.println("partida criada");
        Partida partida =
escolhida.Getpartidas().get(escolhida.Getpartidas().size()-1);
        System.out.println("A partida criada foi: ");
        System.out.println(partida.toString());
        System.out.println("Executando partida...");
        if(partida.getequipa_casa().getJogadoresLista().size() >= 7 &&
partida.getequipa_visitante().getJogadoresLista().size() >= 7){
            System.out.println("A equipa que começa com a posse de bola
é:" + partida.getArbitro().lancarMoedaAoAr(partida.getequipa_casa(),
partida.getequipa_visitante()));
            partida.executarpartida();
        }
        else{
            if(partida.getequipa_casa().getJogadoresLista().size() >= 7 &&
partida.getequipa_visitante().getJogadoresLista().size() < 7){
                System.out.println("A equipa visitante não tem jogadores
suficientes para iniciar a partida.");

partida.getequipa_casa().setNumeroVitorias(partida.getequipa_casa().g
etNumeroVitorias() + 1);

partida.getequipa_casa().setNumeroGolos(partida.getequipa_casa().get
NumeroGolos() + 3);

partida.getequipa_visitante().setNumeroDerrotas(partida.getequipa_visit
ante().getNumeroDerrotas() + 1);

partida.getequipa_visitante().setNumeroGolosSofridos(partida.getequipa
_visitante().getNumeroGolosSofridos() + 3);

partida.setGolos_marcados_casa(partida.getGolos_marcados_casa() +
3);
                partida.setresultado();
                partida.terminou();
```

```
        }
        else if(partida.getequipa_visitante().getJogadoresLista().size() >=
7 && partida.getequipa_casa().getJogadoresLista().size() < 7){
            System.out.println("A equipa da casa não tem jogadores
suficientes para iniciar a partida.");

partida.getequipa_visitante().setNumeroVitorias(partida.getequipa_visita
nte().getNumeroVitorias() + 1);

partida.getequipa_visitante().setNumeroGolos(partida.getequipa_visitant
e().getNumeroGolos() + 3);

partida.getequipa_casa().setNumeroDerrotas(partida.getequipa_casa().
getNumeroDerrotas() + 1);

partida.getequipa_casa().setNumeroGolosSofridos(partida.getequipa_ca
sa().getNumeroGolosSofridos() + 3);

partida.setGolos_sofridos_casa(partida.getGolos_sofridos_casa() + 3);
            partida.setresultado();
            partida.terminou();
        }
        else if(partida.getequipa_visitante().getJogadoresLista().size() <
11 && partida.getequipa_casa().getJogadoresLista().size() < 11){
            System.out.println("Nenhuma das equipas tem jogadores
suficientes para iniciar a partida.");

partida.getequipa_casa().setNumeroEmpates(partida.getequipa_casa().
getNumeroEmpates() + 1);

partida.getequipa_visitante().setNumeroEmpates(partida.getequipa_visit
ante().getNumeroEmpates() + 1);
            partida.setresultado();
            partida.terminou();
        }
    }
    System.out.println("A informação final da partida foi: ");
```

```
        System.out.println(partida.toString());
        }
        else{
            System.out.println("Uma liga tem de ter pelo menos 2 equipas e
um número par de equipas para poder decorrer! Tem ainda de ter pelo
menos um árbitro");
        }

    }

    public void escolha(List<Liga> ligas, List<Equipa>
equipas,List<TreinadorPrincipal> treinadoresPrincipal,List<Jogador>
jogadores,List<Arbitro> arbitros,List<TreinadorAssistente>
treinadoresAssistente,SubMenu submenu){
        for(int i=0; i< ligas.size(); i++){
            for(int j = 0; j < ligas.get(i).getequipas().size(); j++){
                ligas.get(i).getequipas().get(j).definir_jogadores_totais(); //
atualizar lista de jogadores de todas as equipas
            }
        }
        System.out.println("Escolha uma das opções: ");
        String escolha = leitura.nextLine();
        boolean sair=true;
        while(!escolhavalida(escolha)&& sair){
            System.out.println("Opção inválida! Escolha uma opção entre 0 e
9: ");
            escolha = leitura.nextLine();
        }
        int escolhaint = escolha.charAt(0) - 48;
        switch(escolhaint){
            case 1:
                acederinfojogadores(ligas);
                break;
            case 2:
                acederinfotreinadores(ligas);
                break;
            case 3:
```

```
                acederinfoequipa(ligas);
                break;
            case 4:
                criarpartida(ligas);
                break;
            case 5:
                estatisticasEquipa(ligas);
                break;
            case 6:
                mostrartabela(ligas);
                break;
            case 7:
                while(true){
                submenu.opcoes();
                submenu.escolha(ligas, equipas, treinadoresPrincipal,
jogadores, arbitros, treinadoresAssistente);
                }

            case 8:
                System.exit(0);
                break;


        }
    }

    public boolean nomeEquipaExistente(List <Equipa> equipas,String
nomeDaEquipa){

        for(int i=0;i<equipas.size();i++){
            if(equipas.get(i).getNome()==nomeDaEquipa){
                return true;
            }

        }
        return false;
```

```
    }
    public int ligaExisteNomeEquipa(List <Liga> ligas,String nome){

        for(int i=0;i<ligas.size();i++){
            if(nome.equalsIgnoreCase(ligas.get(i).getNome())){
                return i;
            }
        }
        return -1;
    }
    public ArrayList<String> equipasMesmaLiga(List <Liga> ligas,List
<Equipa> equipas,String nome){

        ArrayList<String> temp=new ArrayList();
        for(int i=0;i<equipas.size();i++){
            if(equipas.get(i).getLiga().equalsIgnoreCase(nome)){
                temp.add(equipas.get(i).getNome());
            }

        }
        return temp;
    }
    public ArrayList<String> equipasDaLiga(List <Liga> ligas,String
nome){
        int indiceLiga = ligaExisteNomeEquipa(ligas, nome);
        ArrayList<String> temp=new ArrayList();
        for(int i=0;i<ligas.get(indiceLiga).getequipas().size();i++){
            temp.add(ligas.get(indiceLiga).getequipas().get(i).getNome());
        }
        return temp;
    }
    //funcao que remove o que está em comum entre duas Listas
    public void unicaEquipaQuePossoAdicionar(ArrayList<String>
lista1,ArrayList<String> lista2){
        ArrayList<String> temp=new ArrayList();
        lista2.removeAll(lista1);
    }
```

```
    //Equipas que estao na Liga
    public void associarEquipaLiga(List<Liga> ligas, List<Equipa>
equipas) {
    System.out.println("Digite a Liga: ");
    String nome = leitura.nextLine();
    int indiceLiga = ligaExisteNomeEquipa(ligas, nome);
    if (indiceLiga == -1) {
        System.out.println("Liga não encontrada. Por favor, insira uma liga
válida.");
        return;
    }
    ArrayList<String> listaTodasEquipaLiga = equipasDaLiga(ligas,
nome);
    Collections.sort(listaTodasEquipaLiga);

    ArrayList<String> listaEquipasCriadas = equipasMesmaLiga(ligas,
equipas, ligas.get(indiceLiga).getNome());
    Collections.sort(listaEquipasCriadas);
    unicaEquipaQuePossoAdicionar(listaTodasEquipaLiga,
listaEquipasCriadas);

    System.out.println("Escolha a equipa que quer associar a uma
liga:\n");
    String nomeEquipa;
    for (int i = 0; i < listaEquipasCriadas.size(); i++) {
        System.out.print((i + 1) + "-" + listaEquipasCriadas.get(i) + "\n");
    }
    String equipa = leitura.nextLine();
    int indiceEquipa = Integer.parseInt(equipa) - 1;
    nomeEquipa = listaEquipasCriadas.get(indiceEquipa);
    Equipa equipaAAdicionar = null;

    for (int l = 0; l < equipas.size(); l++) {
        if (equipas.get(l).getLiga().equalsIgnoreCase(nome)) {
            if (nomeEquipa.equalsIgnoreCase(equipas.get(l).getNome())) {
                equipaAAdicionar = equipas.get(l);
```

```
            }
        }
    }

    ligas.get(indiceLiga).addequipa(equipaAAdicionar);

    for (int k = 0; k < ligas.get(indiceLiga).getequipas().size(); k++) {
        System.out.println("\n"+ligas.get(indiceLiga).getequipas().get(k));
    }
}
public void estatisticasEquipa(List <Liga> ligas){
        System.out.println("Digite o nome da equipa: ");
        String nome = leitura.nextLine();
        boolean existe = false;
        for(int j=0; j<ligas.size(); j++){
            for(int i=0; i< ligas.get(j).getequipas().size(); i++){

if(nome.equalsIgnoreCase(ligas.get(j).getequipas().get(i).getNome())){
                    ligas.get(j).getequipas().get(i).DesempenhoMedio();
                    System.out.println(ligas.get(j).getequipas().get(i));
                    existe = true;
                    break;
                }
                else{
                    continue;
                }
            }
        }
        if(!existe){
            System.out.println("Equipa não existe!");
        }
    }
public void mostrartabela(List <Liga> ligas){
        boolean continuar = false;
        Liga escolhida = new Liga();
        while(!continuar ){
            System.out.println("Digite a liga que quer ver a tabela: ");
```

```
            String escolha = leitura.nextLine();
            for(int i = 0; i < ligas.size(); i ++){
                if(ligas.get(i).getNome().equalsIgnoreCase(escolha)){
                    continuar = true;
                    escolhida = ligas.get(i);
                    break;
                }
            }
            if(!continuar){
                System.out.println("Liga não existe");
            }
            else{
                System.out.println("Liga: " + escolhida.getNome());
            }
        }
        System.out.println("Jornada nº" + escolhida.getjornada() + "\n");
        System.out.println("Equipas | NºJogos | NºVitorias | NºDerrotas |
NºEmpates | NºGolos marcados | NºGolos sofridos | Pontos");

System.out.println("----------------------------------------------------------------
--------------------------------");
        for(int j = 0; j<escolhida.getequipas().size(); j++){
            Equipa atual = escolhida.getequipas().get(j);
            System.out.println(atual.getNome() +" | " + atual.n_jogos() + " | "
+ atual.getNumeroVitorias() + " | " + atual.getNumeroDerrotas() + " | " +
atual.getNumeroEmpates() + " | " + atual.getNumeroGolos() + " | " +
atual.getNumeroGolosSofridos() + " | " + atual.pontos());
        }
}
   public void menuCaminho(List<Liga> ligas, List<Equipa> equipas,
    List<TreinadorPrincipal> treinadores,List<Jogador> jogadores,
    List<Arbitro> arbitros,List<TreinadorAssistente>
treinadores_ass,SubMenu submenu,Menu menu)throws IOException{
        boolean validoOpcao = true;
        System.out.println("1.Inserir manualmente\n2.Inserir através de
ficheiros");
        String opcaoLeitura = leitura.nextLine().trim();
```

```
        int opcaoLeituraInt=0;
        while(validoOpcao){
            if(contemLetras(opcaoLeitura)){
                System.out.println("A opção não pode conter letras. Introduza
novamente: ");
                continue;
            }
            else if(opcaoLeitura.trim().isEmpty()){
                 System.out.println("Introduza uma opção válida");
                continue;
            }
            if(opcaoLeitura.trim().length()==1){
                opcaoLeituraInt = Integer.parseInt(opcaoLeitura.trim());
                if(opcaoLeituraInt==1 || opcaoLeituraInt==2){
                    validoOpcao=false;
                    break;
                }
                else{
                    System.out.println("Introduza uma opção válida");
                    continue;
                }
            }
            else{
                System.out.println("Introduza uma opção válida");
                continue;
            }
      }
        if(opcaoLeituraInt==1){
            while(true){
            submenu.opcoes();
            submenu.escolha(ligas, equipas, treinadores, jogadores,
arbitros, treinadores_ass);
            }

        }
        if(opcaoLeituraInt==2){
            String caminho="";
```

```
        Leitura leitura = new Leitura();
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\Port
o.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\Benfi
ca.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\Brag
a.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\CdN
acional.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\Spor
ting.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
        caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Liga-Portuguesa\\Esto
ril.txt";
        leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
```

```
            caminho =
"C:\\Users\\Paulo\\Documents\\NetBeansProjects\\poo.projetofase2\\src\
\main\\java\\com\\mycompany\\poo\\projetofase2\\Arbitros.txt";
            leitura.lerFicheiroArbitro(caminho, ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
            //adicionar todos os arbitros liga portuguesa
            //como so existe 1 liga é adicionado à portuguesa
            if(ligas.size()==1){
                for(int k=0;k<treinadores.size();k++){
                    ligas.get(0).addarbitro(arbitros.get(k));
                }
            }


            while(true){
                menu.opcoes();
                menu.escolha(ligas, equipas, treinadores, jogadores, arbitros,
treinadores_ass, submenu);
            }
        }

}
        public boolean contemNumeros(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    for (int i=0;i<strNumCaracteres;i++){
        if (Character.isDigit(strCaracteres[i])){
            return true;
        }
    }
    return false;
}
    public boolean contemLetras(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    for (int i = 0; i < strNumCaracteres; i++){
        if (Character.isLetter(strCaracteres[i])){
```

```
            return true;
        }
    }
    return false;
}
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;

/**
 *
 * @author Renato
 */
public class Partida {
    private String data;
    private Equipa equipa_casa;
    private Equipa equipa_visitante;
    private Arbitro arbitro;
    private String resultado;


    public int getGolos_sofridos_casa() {
        return golos_sofridos_casa;
    }

    public void setGolos_sofridos_casa(int golos_sofridos_casa) {
        this.golos_sofridos_casa = golos_sofridos_casa;
    }
```

```
public int getGolos_marcados_casa() {
    return golos_marcados_casa;
}

public void setGolos_marcados_casa(int golos_marcados_casa) {
    this.golos_marcados_casa = golos_marcados_casa;
}
private int golos_sofridos_casa = 0;
private int golos_marcados_casa = 0;
private boolean terminada = false;

public void terminou(){
    this.terminada = true;
}

public Equipa getequipa_casa(){
    return this.equipa_casa;
}

public Equipa getequipa_visitante(){
    return this.equipa_visitante;
}

public void setequipa_visitante(Equipa a){
    this.equipa_visitante = a;
}
public void setequipa_casa(Equipa a){
    this.equipa_casa = a;
}

public void setdata(String data){
    this.data = data;
}

public int chance_golo(Equipa a){
    a.definir_tatica();
    a.alterar_moral_treinador();
```

```
        Random aleatorio = new Random();
        int chances_minimas = (14 * a.getTreinadorPrincipal().getMoral()) /
100;
        int chances_maximas = (24 *a.getTreinadorPrincipal().getMoral()) /
100;
        int chances_golo = aleatorio.nextInt((chances_maximas + 1) -
chances_minimas) + chances_minimas;
        int vantagem_casa = aleatorio.nextInt(3-1)+1;
        if(this.equipa_casa.equals(a)){
            chances_golo = chances_golo + vantagem_casa;
        }
        return chances_golo;
    }

    public int probabilidade_marcar_golo(Equipa a){
        int probabilidade = 0;
        Random aleatorio = new Random();
        for(int i = 0; i<a.getJogadoresLista().size(); i++){
            if(a.getJogadoresLista().get(i).getPosJogador().equals("GR")){
                probabilidade = probabilidade + (aleatorio.nextInt(2-0)+0);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("DEFESA")){
                probabilidade = probabilidade + (aleatorio.nextInt(8-0)+0);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("MEDIO")){
                probabilidade = probabilidade + (((100 +
a.getJogadoresLista().get(i).getOverall()) * (aleatorio.nextInt(31-10)+10))
/ 100);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("ATACANTE")){
                probabilidade = probabilidade + (((100 +
a.getJogadoresLista().get(i).getOverall()) * (aleatorio.nextInt(61-31)+31))
/ 100);
            }
```

```
        }
        return probabilidade;
    }

    public void setarbitro(Arbitro a){
        this.arbitro = a;
    }

    public int probabilidade_defender_golo(Equipa a){
        int probabilidade = 0;
        Random aleatorio = new Random();
        for(int i = 0; i<a.getJogadoresLista().size(); i++){

if(a.getJogadoresLista().get(i).getPosJogador().equals("ATACANTE")){
                probabilidade = probabilidade + (aleatorio.nextInt(2-0)+0);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("MEDIO")){
                probabilidade = probabilidade + (aleatorio.nextInt(8-0)+0);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("DEFESA")){
                probabilidade = probabilidade + (((100 +
a.getJogadoresLista().get(i).getOverall()) * (aleatorio.nextInt(31-10)+10))
/ 100);
            }
            else
if(a.getJogadoresLista().get(i).getPosJogador().equals("GR")){
                probabilidade = probabilidade + (((100 +
a.getJogadoresLista().get(i).getOverall()) * (aleatorio.nextInt(61-31)+31))
/ 100);
            }
        }
        return probabilidade;
    }

    public void golos(){
```

```
        int chance_golo_casa = chance_golo(this.equipa_casa);
        System.out.println("Chances casa: " + chance_golo_casa);
        int chance_golo_visitante = chance_golo(this.equipa_visitante);
        System.out.println("Chance fora: " + chance_golo_visitante);
        Random aleatorio = new Random();
        for(int i = 0; i < chance_golo_casa; i ++){
            if(probabilidade_marcar_golo(this.equipa_casa) >
probabilidade_defender_golo(this.equipa_visitante)){
                this.golos_marcados_casa = this.golos_marcados_casa + 1;
                int chance_marcar_golo = -1;
                int chance_assistir = -1;
                Jogador marcador = new Jogador("", 0, "", 0);
                Jogador assistente = new Jogador("", 0, "", 0);
                //fazer uma chance pra saber qual jogador da casa marcou o
golo
                for(int j=0; j < this.equipa_casa.getJogadoresLista().size();
j++){
                    Jogador jogador_atual =
this.equipa_casa.getJogadoresLista().get(j);
                    int chance_marcar_golo_atual = 0;
                    if(jogador_atual.getPosJogador().equalsIgnoreCase("GR")){
                        chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(2-0)+0);
                    }
                    else if(jogador_atual.getPosJogador().equals("DEFESA")){
                        chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(8-0)+0);
                    }
                    else if(jogador_atual.getPosJogador().equals("MEDIO")){
                        chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(50-0)+0);
                    }
                    else
if(jogador_atual.getPosJogador().equals("ATACANTE")){
                        chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(100-0)+0);
                    }
```

```
                if(chance_marcar_golo_atual > chance_marcar_golo){
                    chance_marcar_golo = chance_marcar_golo_atual;
                    marcador = jogador_atual;
                }
                else if(chance_marcar_golo_atual > chance_assistir){
                    chance_assistir = chance_marcar_golo_atual;
                    assistente = jogador_atual;
                }
            }

marcador.getInfo_ataque().setGolos(marcador.getInfo_ataque().getGolo
s() + 1);

assistente.getInfo_ataque().setAssistencias(assistente.getInfo_ataque().
getAssistencias() + 1);
        }
        else{ //nao ha golo - ver quem defendeu
            int chance_defender_golo = -1;
            Jogador defensor = new Jogador("", 0, "", 0);
            for(int j=0; j < this.equipa_visitante.getJogadoresLista().size();
j++){
                Jogador jogador_atual =
this.equipa_visitante.getJogadoresLista().get(j);
                int chance_defender_golo_atual = 0;
                if(jogador_atual.getPosJogador().equals("GR")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(100-0)+0);
                }
                else if(jogador_atual.getPosJogador().equals("DEFESA")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(50-0)+0);
                }
                else if(jogador_atual.getPosJogador().equals("MEDIO")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(30-0)+0);
                }
```

```
                else
if(jogador_atual.getPosJogador().equals("ATACANTE")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(2-0)+0);
                }
                if(chance_defender_golo_atual > chance_defender_golo){
                    chance_defender_golo = chance_defender_golo_atual;
                    defensor = jogador_atual;
                }
            }

defensor.getInfo_defesa().setGolos_impedidos(defensor.getInfo_defesa(
).getGolos_impedidos() + 1);

            }
        }
        for(int i = 0; i < chance_golo_visitante; i ++){
            if(probabilidade_marcar_golo(this.equipa_visitante) >
probabilidade_defender_golo(this.equipa_casa)){
                this.golos_sofridos_casa = this.golos_sofridos_casa + 1;
                int chance_marcar_golo = -1;
                int chance_assistir = -1;
                Jogador marcador = new Jogador("", 0, "", 0);
                Jogador assistente = new Jogador("", 0, "", 0);
                //fazer uma chance pra saber qual jogador da casa marcou o
golo
                for(int j=0; j < this.equipa_visitante.getJogadoresLista().size();
j++){
                    Jogador jogador_atual =
this.equipa_visitante.getJogadoresLista().get(j);
                    int chance_marcar_golo_atual = 0;
                    if(jogador_atual.getPosJogador().equals("GR")){
                        chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(2-0)+0);
                    }
                    else if(jogador_atual.getPosJogador().equals("DEFESA")){
```

```
                    chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(8-0)+0);
                }
                else if(jogador_atual.getPosJogador().equals("MEDIO")){
                    chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(50-0)+0);
                }
                else
if(jogador_atual.getPosJogador().equals("ATACANTE")){
                    chance_marcar_golo_atual = jogador_atual.getOverall()
+ (aleatorio.nextInt(100-0)+0);
                }
                if(chance_marcar_golo_atual > chance_marcar_golo){
                    chance_marcar_golo = chance_marcar_golo_atual;
                    marcador = jogador_atual;
                }
                else if(chance_marcar_golo_atual > chance_assistir){
                    chance_assistir = chance_marcar_golo_atual;
                    assistente = jogador_atual;
                }
            }

marcador.getInfo_ataque().setGolos(marcador.getInfo_ataque().getGolo
s() + 1);

assistente.getInfo_ataque().setAssistencias(assistente.getInfo_ataque().
getAssistencias() + 1);
        }
        else{ //nao ha golo - ver quem defendeu
            int chance_defender_golo = -1;
            Jogador defensor = new Jogador("", 0, "", 0);
            for(int j=0; j < this.equipa_casa.getJogadoresLista().size();
j++){
                Jogador jogador_atual =
this.equipa_casa.getJogadoresLista().get(j);
                int chance_defender_golo_atual = 0;
                if(jogador_atual.getPosJogador().equals("GR")){
```

```
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(100-0)+0);
                }
                else if(jogador_atual.getPosJogador().equals("DEFESA")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(50-0)+0);
                }
                else if(jogador_atual.getPosJogador().equals("MEDIO")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(30-0)+0);
                }
                else
if(jogador_atual.getPosJogador().equals("ATACANTE")){
                    chance_defender_golo_atual =
jogador_atual.getOverall() + (aleatorio.nextInt(2-0)+0);
                }
                if(chance_defender_golo_atual > chance_defender_golo){
                    chance_defender_golo = chance_defender_golo_atual;
                    defensor = jogador_atual;
                }

            }

defensor.getInfo_defesa().setGolos_impedidos(defensor.getInfo_defesa(
).getGolos_impedidos() + 1);
            }
            }
        }
    public void atualizar_infos_equipas(){ // alterar golos das equipas e n
de vitorias...
            if(this.golos_marcados_casa > this.golos_sofridos_casa){

this.equipa_casa.setNumeroVitorias(this.equipa_casa.getNumeroVitoria
s() + 1);

this.equipa_visitante.setNumeroDerrotas(this.equipa_visitante.getNumer
oDerrotas() + 1);
```

```
        }
        else if(this.golos_marcados_casa < this.golos_sofridos_casa){

this.equipa_visitante.setNumeroVitorias(this.equipa_visitante.getNumero
Vitorias() + 1);

this.equipa_casa.setNumeroDerrotas(this.equipa_casa.getNumeroDerro
tas() + 1);
        }
        else if(this.golos_marcados_casa == this.golos_sofridos_casa){

this.equipa_casa.setNumeroEmpates(this.equipa_casa.getNumeroEmp
ates() + 1);

this.equipa_visitante.setNumeroEmpates(this.equipa_visitante.getNumer
oEmpates() + 1);
        }

this.equipa_casa.setNumeroGolos(this.equipa_casa.getNumeroGolos()
+ this.golos_marcados_casa);

this.equipa_casa.setNumeroGolosSofridos(this.equipa_casa.getNumero
GolosSofridos() + this.golos_sofridos_casa);

this.equipa_visitante.setNumeroGolos(this.equipa_visitante.getNumeroG
olos() + this.golos_sofridos_casa);

this.equipa_visitante.setNumeroGolosSofridos(this.equipa_visitante.get
NumeroGolosSofridos() + this.golos_marcados_casa);
    }
    public void estatisticas_defensivas(){
        Random aleatorio = new Random();
        for(int i = 0; i < this.equipa_casa.getJogadoresLista().size(); i ++){
            Jogador atual = this.equipa_casa.getJogadoresLista().get(i);
            if(atual.getPosJogador().equals("GR")){
```

```
atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(3-0)+0));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(2-0)+0));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(2-0)+0));
        }
        else if(atual.getPosJogador().equals("DEFESA")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(20-6)+6));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(10-3)+3));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(20-6)+6));
        }
        else if(atual.getPosJogador().equals("MEDIO")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(12-2)+2));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(7-1)+1));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(12-2)+2));
        }
        else if(atual.getPosJogador().equals("ATACANTE")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(5-0)+0));
```

```
atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(2-0)+0));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(5-0)+0));
            }
        }
    for(int i = 0; i < this.equipa_visitante.getJogadoresLista().size(); i ++){
            Jogador atual = this.equipa_visitante.getJogadoresLista().get(i);
            if(atual.getPosJogador().equals("GR")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(3-0)+0));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(2-0)+0));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(2-0)+0));
            }
            else if(atual.getPosJogador().equals("DEFESA")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(20-6)+6));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(10-3)+3));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(20-6)+6));
            }
            else if(atual.getPosJogador().equals("MEDIO")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(12-2)+2));
```

```
atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(7-1)+1));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(12-2)+2));
            }
            else if(atual.getPosJogador().equals("ATACANTE")){

atual.getInfo_defesa().setBolas_recuperadas(atual.getInfo_defesa().get
Bolas_recuperadas() + (aleatorio.nextInt(5-0)+0));

atual.getInfo_defesa().setCarrinhos_certos(atual.getInfo_defesa().getCa
rrinhos_certos() + (aleatorio.nextInt(2-0)+0));

atual.getInfo_defesa().setCortes_certos(atual.getInfo_defesa().getCortes
_certos() + (aleatorio.nextInt(5-0)+0));
            }
        }
    }

    public void estatisticas_ofensivas(){
        Random aleatorio = new Random();
        for(int i = 0; i < this.equipa_casa.getJogadoresLista().size(); i ++){
            Jogador atual = this.equipa_casa.getJogadoresLista().get(i);
            if(atual.getPosJogador().equals("GR")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPass
es_certos() + (aleatorio.nextInt(8-2)+2));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDrible
s_certos() + (aleatorio.nextInt(2-0)+0));
            }
            else if(atual.getPosJogador().equals("DEFESA")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPass
es_certos() + (aleatorio.nextInt(15-5)+5));
```

```
atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(5-0)+0));
            }
            else if(atual.getPosJogador().equals("MEDIO")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(30-10)+10));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(10-2)+2));
            }
            else if(atual.getPosJogador().equals("ATACANTE")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(25-8)+8));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(20-5)+5));
            }

    }
    for(int i = 0; i < this.equipa_visitante.getJogadoresLista().size(); i ++){
            Jogador atual = this.equipa_visitante.getJogadoresLista().get(i);
            if(atual.getPosJogador().equals("GR")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(8-2)+2));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(2-0)+0));
            }
            else if(atual.getPosJogador().equals("DEFESA")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(15-5)+5));
```

```
atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(5-0)+0));
            }
            else if(atual.getPosJogador().equals("MEDIO")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(30-10)+10));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(10-2)+2));
            }
            else if(atual.getPosJogador().equals("ATACANTE")){

atual.getInfo_ataque().setPasses_certos(atual.getInfo_ataque().getPasses_certos() + (aleatorio.nextInt(25-8)+8));

atual.getInfo_ataque().setDribles_certos(atual.getInfo_ataque().getDribles_certos() + (aleatorio.nextInt(20-5)+5));
            }

    }
    }

    public boolean demasiados_cartoes_casa(){
        Random aleatorio = new Random();
        int n_expulsos = 0;

        for(int i = 0; i < this.equipa_casa.getJogadoresLista().size(); i++){
            Jogador atual = this.equipa_casa.getJogadoresLista().get(i);
            int chance_cartao = aleatorio.nextInt(atual.getagressividade() - 0) +0;
            if(chance_cartao > 75){
                System.out.println("Jogador expulso: " + atual.getNome());

atual.getInfo_defesa().setCartoesvermelhos(atual.getInfo_defesa().getCartoesvermelhos() + 1);
```

```
                atual.setdias_indisponiveis(1);
                this.equipa_casa.getJogadores_indisponiveis().add(atual);
                this.equipa_casa.getJogadoresLista().remove(atual);
                n_expulsos = n_expulsos + 1;
            }
            else if(chance_cartao > 50 && chance_cartao <= 75){
                System.out.println("Jogador levou amarelo: " +
atual.getNome());

atual.getInfo_defesa().setCartoesamarelos(atual.getInfo_defesa().getCa
rtoesamarelos() + 1);
            }
        }
        if(n_expulsos > 4){
            return true;
        }
        else{
            return false;
        }
    }
    public boolean demasiados_cartoes_fora(){
        Random aleatorio = new Random();
        int n_expulsos = 0;
        for(int i = 0; i < this.equipa_visitante.getJogadoresLista().size();
i++){
            Jogador atual = this.equipa_visitante.getJogadoresLista().get(i);
            int chance_cartao = aleatorio.nextInt(atual.getagressividade() -
0) +0;
            if(chance_cartao > 75){
                System.out.println("Jogador expulso: " + atual.getNome());

atual.getInfo_defesa().setCartoesvermelhos(atual.getInfo_defesa().getC
artoesvermelhos() + 1);
                this.equipa_visitante.getJogadores_indisponiveis().add(atual);
                this.equipa_visitante.getJogadoresLista().remove(atual);
                n_expulsos = n_expulsos + 1;
            }
```

```
            else if(chance_cartao > 50 && chance_cartao <= 75){
                System.out.println("Jogador levou amarelo: " +
atual.getNome());

atual.getInfo_defesa().setCartoesamarelos(atual.getInfo_defesa().getCa
rtoesamarelos() + 1);
            }
        }
        if(n_expulsos > 4){
            return true;
        }
        else{
            return false;
        }
    }

    public void lesoes(String data){
        Random aleatorio = new Random();
        for(int i = 0; i<this.equipa_visitante.getJogadoresLista().size();i++){
            Jogador atual = this.equipa_visitante.getJogadoresLista().get(i);
            int chance_lesao = aleatorio.nextInt(100 - 0) + 0;
            if(chance_lesao == 99){
                System.out.println("O jogador "+ atual.getNome() + "
lesionou-se.");
                int gravidade_lesao = aleatorio.nextInt(6-1)+ 1;
                atual.setdias_indisponiveis(gravidade_lesao);
                this.equipa_visitante.getJogadores_indisponiveis().add(atual);
                this.equipa_visitante.getJogadoresLista().remove(atual);
                atual.novaLesaoaleatoria(data);
            }
        }
        for(int i = 0; i<this.equipa_casa.getJogadoresLista().size();i++){
            Jogador atual = this.equipa_casa.getJogadoresLista().get(i);
            int chance_lesao = aleatorio.nextInt(100 - 0) + 0;
            if(chance_lesao > 94){
                System.out.println("O jogador "+ atual.getNome() + "
lesionou-se.");
```

```
                int gravidade_lesao = aleatorio.nextInt(6-1)+ 1;
                atual.setdias_indisponiveis(gravidade_lesao);
                this.equipa_casa.getJogadores_indisponiveis().add(atual);
                this.equipa_casa.getJogadoresLista().remove(atual);
                atual.novaLesaoaleatoria(data);
            }
        }
    }

    public void executarpartida(){
        if(!this.terminada){
            if(!demasiados_cartoes_casa() && !demasiados_cartoes_fora()){
            golos();
            atualizar_infos_equipas();
            estatisticas_defensivas();
            estatisticas_ofensivas();
            lesoes(this.data);
            this.terminada = true;
            }
            else if(demasiados_cartoes_fora()){
                System.out.println("A equipa visitante teve demasiadas
expulsões!");
                lesoes(this.data);
                this.golos_marcados_casa = 3;
                this.golos_sofridos_casa = 0;
                setresultado();

this.equipa_casa.setNumeroGolos(this.equipa_casa.getNumeroGolos()
+ 3);

this.equipa_casa.setNumeroVitorias(this.equipa_casa.getNumeroVitoria
s() + 1);

this.equipa_visitante.setNumeroGolosSofridos(this.equipa_visitante.get
NumeroGolosSofridos() + 3);
```

```
this.equipa_visitante.setNumeroDerrotas(this.equipa_visitante.getNumer
oDerrotas() + 1);
            }
            else if(demasiados_cartoes_casa()){
                lesoes(this.data);
                System.out.println("A equipa da casa teve demasiadas
expulsões!");
                this.golos_marcados_casa = 0;
                this.golos_sofridos_casa = 3;
                setresultado();

this.equipa_visitante.setNumeroGolos(this.equipa_visitante.getNumeroG
olos() + 3);

this.equipa_visitante.setNumeroVitorias(this.equipa_visitante.getNumero
Vitorias() + 1);

this.equipa_casa.setNumeroGolosSofridos(this.equipa_casa.getNumero
GolosSofridos() + 3);

this.equipa_casa.setNumeroDerrotas(this.equipa_casa.getNumeroDerro
tas() + 1);

            }
        }
        else{
            System.out.println("Esta partida já foi disputada anteriormente!");
        }
    }

    public void setresultado(){
        this.resultado = this.golos_marcados_casa + "-" +
this.golos_sofridos_casa;
    }
    public String toString(){
        if(!this.terminada){
```

```
            return "Data: " + this.data + "\nEquipa da casa: " +
this.equipa_casa.getNome() + "\nEquipa visitante: " +
this.equipa_visitante.getNome()+ "\nArbitro: " + this.arbitro.getNome();
        }
        else{
            setresultado();
            return "Data: " + this.data + "\nEquipa da casa: " +
this.equipa_casa.getNome() + "\nEquipa visitante: " +
this.equipa_visitante.getNome()+ "\nArbitro: " + this.arbitro.getNome() +
"\nResultado:" + this.resultado;
        }
    }
    public Arbitro getArbitro(){
        return this.arbitro;
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;

/**
 *
 * @author Renato
 */
public class Pessoa {
    protected String nome;
    protected int idade;
    protected int overall = 0;

    public int getOverall() {
        return overall;
```

```
    }

    public void setOverall(int overall) {
        this.overall = overall;
    }


    public String getNome(){
        return nome;
    }
    public void setNome(String nome){
        this.nome=nome;
    }
    public int getIdade(){
        return idade;
    }

    public void setIdade(int idade){
        this.idade=idade;
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 */

package com.mycompany.poo.projetofase2;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Paulo
```

```
 */
public class PooProjetofase2 {

    public static void main(String[] args) throws IOException {
        List <Equipa> equipas = new ArrayList();
        List <TreinadorPrincipal> treinadores = new ArrayList();
        List <TreinadorAssistente> treinadores_ass = new ArrayList();
        List <Jogador> jogadores = new ArrayList();
        List <Arbitro> arbitros = new ArrayList();
        List <Liga> ligas = new ArrayList();
        Menu menu = new Menu();
        SubMenu submenu = new SubMenu();
        menu.menuCaminho(ligas, equipas, treinadores, jogadores,
arbitros, treinadores_ass,submenu,menu);
//
//        if(args.length==0){
//        Menu menu = new Menu();
//        //menu.opcoes();
////        Equipa BENFICA = new Equipa();
////        BENFICA.setLiga("Portuguesa");
////        BENFICA.setNome("Benfica");
////        equipas.add(BENFICA);
////        BENFICA.setTreinador("Lucas",90);
////        BENFICA.getTreinador().setNome("ANIBAL");
////
BENFICA.getTreinador().ADDEspecializacoes(BENFICA.getTreinador().
GetEspecializacoes(),"sábio");
////
BENFICA.getTreinador().ADDTaticaPref(BENFICA.getTreinador().GetTat
icaspref(),"4-2-2");
////        int temp=BENFICA.getJogadoresLista().size();
////        BENFICA.getJogadoresLista().add(new
Jogador("Paulo",10,"S",10));
////        BENFICA.getJogadoresLista().add(new Jogador("S",10,"S",10));
////
//        Liga Portuguesa = new Liga("Portuguesa");
//        Liga Espanhola = new Liga("Espanhola");
```

```
////        Portuguesa.addequipa(BENFICA);
//        Equipa porto = new Equipa("porto","Portuguesa");
//        equipas.add(porto);
//        Equipa benfica = new Equipa("benfica","Portuguesa");
//        equipas.add(benfica);
////        Equipa braga = new Equipa("braga","Portuguesa");
////        equipas.add(braga);
////        Equipa famalicao = new Equipa("famalicao","Portuguesa");
////        equipas.add(famalicao);
////        Equipa vitoria = new Equipa("vitoria","Portuguesa");
////        equipas.add(vitoria);
//        Equipa barcelona = new Equipa("barcelona","Espanhola");
//        equipas.add(barcelona);
////        Equipa valencia = new Equipa("valencia","Espanhola");
////        equipas.add(valencia);
//        Espanhola.addequipa(barcelona);
//
//        //menu.escolha(ligas,equipas);
//        jogadores.add(new Jogador("Paulo",21,"GK",50));
//        jogadores.add(new Jogador("Lucas",21,"DEFESA",50));
//        jogadores.add(new Jogador("Renato",21,"MEDIO",50));
//        jogadores.add(new Jogador("Gabriel",21,"ATACANTE",50));
//        jogadores.add(new Jogador("Manel",21,"GK",50));
//        jogadores.add(new Jogador("Ribeiro",21,"DEFESA",50));
//        jogadores.add(new Jogador("Lopes",21,"MEDIO",50));
//        jogadores.add(new Jogador("Antonio",21,"ATACANTE",50));
//        jogadores.add(new Jogador("Paulino",21,"MEDIO",50));
//        jogadores.add(new Jogador("Paula",21,"ATACANTE",50));
//        jogadores.add(new Jogador("Jubas",21,"MEDIO",50));
//        jogadores.add(new Jogador("Pilantra",21,"ATACANTE",50));
//
//        //teste
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Paulo",21,"GR",99, 10) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Paul",21,"GR",1, 99) );
```

```
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Pau",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Pa",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("P",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Jaulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Gaulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Faulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Daulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Saulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Aaulo",21,"DEFESA",99, 10) );
//        porto.addJogadores(porto.getJogadoresLista(),new
Jogador("Qaulo",21,"DEFESA",99, 10) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Oaulo",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Yonto",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Taulo",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Raulo",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Eaulo",21,"DEFESA",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Waulo",21,"MEDIO",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Baulo",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("Maulo",21,"ATACANTE",50, 99) );
```

```
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 99) );
//        benfica.addJogadores(benfica.getJogadoresLista(),new
Jogador("ulo",21,"ATACANTE",50, 99) );
//        Arbitro novo = new Arbitro();
//        TreinadorPrincipal roger = new TreinadorPrincipal("Roger", 40,
99);
//        TreinadorPrincipal Conceicao = new
TreinadorPrincipal("Conceicao", 40, 79);
//        benfica.setTreinadorPrincipal(roger);
//        porto.setTreinadorPrincipal(Conceicao);
//        novo.setExperiencia(99);
//        novo.setIdade(30);
//        novo.setNome("Gay");
//        Portuguesa.addarbitro(novo);
//        Portuguesa.addequipa(porto);
//        Portuguesa.addequipa(benfica);
//        Equipa sporting = new Equipa("Sporting", "Portuguesa");
//        TreinadorPrincipal Amorim = new TreinadorPrincipal("Amorim",
30, 50);
//        sporting.setTreinadorPrincipal(Amorim);
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//                sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
```

```
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"ATACANTE",50, 50) );
//        sporting.addJogadores(sporting.getJogadoresLista(),new
Jogador("aulo",21,"GR",50, 50) );
//
//        //Portuguesa.addequipa(sporting);
//        ligas.add(Espanhola);
//        ligas.add(Portuguesa);
//        //
//        SubMenu subMenu = new SubMenu();
//        while(true){
//        subMenu.opcoes();
//        subMenu.escolha(ligas, equipas, treinadores, jogadores, arbitros,
treinadores_ass);
//        }
//    }
//        else if(args.length==1){


//        SubMenu subMenu = new SubMenu();
//        while(true){
//        subMenu.opcoes();
//        subMenu.escolha(ligas, equipas, treinadores, jogadores, arbitros,
treinadores_ass);
//        }
//            //leitura.lerFicheiroEquipa(caminho,ligas, equipas, treinadores,
jogadores, arbitros, treinadores_ass);
//        }
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;

import java.util.List;
import java.util.ArrayList;
/**
 *
 * @author Paulo
 */
public class SubMenu extends Menu implements Comum {
    @Override
    public void opcoes(){
        System.out.println(
                "1. Registar um jogador \n2. Registar um treinador principal \n"
                +"3. Registar um treinador assistente \n"+"4. Registar uma
equipa \n"+"5. Registar um arbítro \n"
                +"6. Ir para o menu\n" + "7. Sair");
    }

    @Override
    public boolean escolhavalida(String a) {
        return super.escolhavalida(a);
    }

    public void escolha(List<Liga> ligas, List<Equipa>
equipas,List<TreinadorPrincipal> treinadoresPrincipal,List<Jogador>
jogadores,List<Arbitro> arbitros,List<TreinadorAssistente>
treinadoresAssistente) {
        System.out.println("Escolha uma das opções: ");
        String escolha = leitura.nextLine().trim();
        boolean sair=true;
        while(!escolhavalida(escolha)&&sair){
            System.out.println("Opção inválida! Escolha uma opção entre 0 e
9: ");
            escolha = leitura.nextLine().trim();
```

```
        }
        int escolhaint = escolha.charAt(0) - 48;
        switch(escolhaint){
            case 1:
                registarJogador(jogadores);
                break;
            case 2:
                registarTreinadorPrincipal(treinadoresPrincipal);
                break;
            case 3:
                registarTreinadorAssistente(treinadoresAssistente);
                break;
            case 4:

registarEquipa(ligas,jogadores,equipas,treinadoresPrincipal,treinadores
Assistente);
                break;
            case 5:
                registarArbitro(arbitros);
                break;
            case 6:
                //Menu();
                voltarAoMenu(ligas, equipas, treinadoresPrincipal, jogadores,
arbitros, treinadoresAssistente);
                break;
            case 7:
                System.exit(0);
                break;
        }
    }
    //Jogadores com o mesmo nome nao posso adicionar
    public boolean percorrerJogadores(List<Jogador> jogadores,String
nomeJogador){
        for(int i=0;i<jogadores.size();i++){
            if(nomeJogador.equalsIgnoreCase(jogadores.get(i).getNome())){
                return true;
            }
```

```
        }
        return false;
    }

    public boolean percorrerTreinadoresPrincipal(List<TreinadorPrincipal>
treinadores,String nomeTreinador){
        for(int i=0;i<treinadores.size();i++){

if(nomeTreinador.equalsIgnoreCase(treinadores.get(i).getNome())){
                return true;
            }
        }
        return false;
    }
    public boolean
percorrerTreinadoresAssistente(List<TreinadorAssistente>
treinadores,String nomeTreinador){
        for(int i=0;i<treinadores.size();i++){

if(nomeTreinador.equalsIgnoreCase(treinadores.get(i).getNome())){
                return true;
            }
        }
        return false;
    }

    public boolean percorrerEquipas(List<Equipa> equipas,String
nomeEquipa){
        for(int i=0;i<equipas.size();i++){
            if(nomeEquipa.equalsIgnoreCase(equipas.get(i).getNome())){
                return true;
            }
        }
        return false;
    }
    //true especializacao já existe
```

```
    public boolean
percorrerTreinadoresEspecializacoes(TreinadorPrincipal treinador,
String especializacao) {
    List<String> especializacoes=treinador.GetEspecializacoes();
    int tamanhoListaEspecializacoes=especializacoes.size();

    if(tamanhoListaEspecializacoes > 0){
        for(int i=0;i<tamanhoListaEspecializacoes;i++){
            if(especializacoes.get(i).equalsIgnoreCase(especializacao)){
                return true;
            }

        }
    }
    return false;
}
    public boolean percorrerTreinadoresTaticas(TreinadorPrincipal
treinador, String tatica) {
    List<String> taticas=treinador.GetTaticaspref();
    int tamanhoListaTaticas=taticas.size();

    if(tamanhoListaTaticas > 0){
        for(int i=0;i<tamanhoListaTaticas;i++){
            if(taticas.get(i).equalsIgnoreCase(tatica)){
                return true;
            }

        }
    }
    return false;
}

    public boolean contemNumeros(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    for (int i=0;i<strNumCaracteres;i++){
        if (Character.isDigit(strCaracteres[i])){
```

```
            return true;
        }
    }
    return false;
}
    public boolean contemLetras(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    for (int i = 0; i < strNumCaracteres; i++){
        if (Character.isLetter(strCaracteres[i])){
            return true;
        }
    }
    return false;
}

    public String posicaoAdvemOpcao(int posicaoJogadorInt){
        switch(posicaoJogadorInt){
            case 1:
                return "GR";
            case 2:
                return "DEFESA";
            case 3:
                return "MEDIO";
            case 4:
                return "ATACANTE";
        }
        return "Essa posição não existe";
    }
    public boolean idadeJogador(int idadeJogadorInt){
        if(idadeJogadorInt>=16 && idadeJogadorInt<=50){
            return true;
        }
        else{
            return false;
        }
    }
```

```
    public boolean agressividadeJogador(int agressividadeJogadorInt){
        if(agressividadeJogadorInt>=0 && agressividadeJogadorInt<=100){
            return true;
        }
        else{
            return false;
        }
    }

    public boolean taticavalida(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    int somaCaracteres=0;
    if(strNumCaracteres==3){
        for (int i = 0; i < strNumCaracteres; i++){
        somaCaracteres+=Character.getNumericValue(strCaracteres[i]);
        }
        if(somaCaracteres==10){
            return true;
        }
        else{
            return false;
        }
    }
    else{
        return false;
    }

}
    public String adicionarSeparacaoTatica(String str){
    char[] strCaracteres = str.toCharArray();
    int strNumCaracteres = strCaracteres.length;
    String taticaResultante = String.valueOf(strCaracteres[0]);
        for(int i=1;i<strNumCaracteres;i++){
            taticaResultante += "-" + strCaracteres[i];
        }
        return taticaResultante;
```

```
    }

    public List<Jogador> mostrarJogadoresPos(String pos,List<Jogador>
jogadores){
        List<Jogador> jogadoresDisp=new ArrayList();
        for(int i=0;i<jogadores.size();i++){
            if(jogadores.get(i).getPosJogador().equalsIgnoreCase(pos)){
                if(jogadores.get(i).getEquipa().equals("")){
                    jogadoresDisp.add(jogadores.get(i));
                }

            }
        }
        return jogadoresDisp;
    }
    public List<Jogador> mostrarJogadoresPos(List<Jogador>
jogadores){
        List<Jogador> jogadoresDisp=new ArrayList();
        for(int i=0;i<jogadores.size();i++){
            if(!(jogadores.get(i).getPosJogador().equals("GR"))){
                if(jogadores.get(i).getEquipa().equals("")){
                    jogadoresDisp.add(jogadores.get(i));
                }

            }
        }
        return jogadoresDisp;
    }
    public List<TreinadorPrincipal>
mostrarTreinadoresPrincipal(List<TreinadorPrincipal> treinadores){
       List<TreinadorPrincipal> treinadoresDisp=new ArrayList();
       for(int i=0;i<treinadores.size();i++){
           if(treinadores.get(i).getEquipaAtreinar().equals("")){
               treinadoresDisp.add(treinadores.get(i));
           }
       }
       return treinadoresDisp;
```

```
    }
    public List<TreinadorAssistente>
mostrarTreinadoresAssistente(List<TreinadorAssistente> treinadores){
        List<TreinadorAssistente> treinadoresDisp=new ArrayList();
        for(int i=0;i<treinadores.size();i++){
            if(treinadores.get(i).getEquipaAtreinar().equals("")){
                treinadoresDisp.add(treinadores.get(i));
            }
        }
        return treinadoresDisp;
    }

    public void registarJogador(List<Jogador> jogadores){
        Jogador jogadorAAdicionar = new Jogador();
        System.out.println("Introduza o nome do Jogador: ");
        boolean validoNome=true;

        while(validoNome){
            String nomeJogador = leitura.nextLine().trim();
            if(nomeJogador.trim().isEmpty()){
                System.out.println("Introduza um nome válido");
                continue;
            }
            else if(percorrerJogadores(jogadores,nomeJogador)){
                System.out.println("Esse jogador já existe, introduza outro");
                continue;
            }
            else if(contemNumeros(nomeJogador)){
                System.out.println("O nome do jogador não pode ter números,
introduza um nome válido");
                continue;
            }
            else{
                validoNome=false;
                jogadorAAdicionar.setNome(nomeJogador.trim());

            }
```

```
        }

            boolean validoIdade=true;
            System.out.println("Introduza a idade do jogador entre 16 anos e
50 anos: ");
            while(validoIdade){
                String idadeJogadorLeitura = leitura.nextLine().trim();
                if (contemLetras(idadeJogadorLeitura)) {
                    System.out.println("A idade não pode conter letras.
Introduza novamente: ");
                    continue;
                }
                else if(idadeJogadorLeitura.trim().isEmpty()){
                    System.out.println("Introduza uma idade válida");
                    continue;
                }
                int idadeJogador=0;
                if(idadeJogadorLeitura.length()==1 ||
idadeJogadorLeitura.length()==2){
                    idadeJogador = Integer.parseInt(idadeJogadorLeitura);
                    if(idadeJogador >= 16 && idadeJogador <= 50){
                        validoIdade = false;
                        jogadorAAdicionar.setIdade(idadeJogador);
                        break;
                    }
                    else{
                        System.out.println("Introduza uma idade válida entre 16
anos e 50 anos: ");
                        continue;
                    }
                }
                else{
                    System.out.println("Introduza uma idade válida entre 16
anos e 50 anos: ");
                    continue;
                }
```

```
        }

      boolean validoPos=true;
      System.out.println("Introduza a sua posição:\n" + "1. GR\n2.
DEFESA\n3. MEDIO\n4. ATACANTE");
         while(validoPos){
            String posicaoJogadorLeitura = leitura.nextLine().trim();
            if(contemLetras(posicaoJogadorLeitura)){
               System.out.println("A posição deve ser um número.
Introduza novamente: ");
               continue;
            }
            else if(posicaoJogadorLeitura.trim().isEmpty()){
               System.out.println("Introduza uma opção válida");
               continue;
            }
            int posicaoJogadorInt=0;
            if(posicaoJogadorLeitura.length()==1){
               posicaoJogadorInt =
Integer.parseInt(posicaoJogadorLeitura);
               String posicaoJogador =
posicaoAdvemOpcao(posicaoJogadorInt);
               if(posicaoJogador.equalsIgnoreCase("Essa posição não
existe")){
                  System.out.println("Introduza uma opção válida:\n1.
GR\n2. DEFESA\n3. MEDIO\n4. ATACANTE");
               }
               else{
                  validoPos = false;

jogadorAAdicionar.setPosJogador(posicaoJogador.trim());
                  break;
               }
            }
            else{
               System.out.println("Introduza uma opção válida:\n1. GR\n2.
DEFESA\n3. MEDIO\n4. ATACANTE");
```

```
                continue;
            }

        }
        boolean validoAgressividade=true;
        System.out.println("Introduza o nivel de agressividade entre 0 e
100");
            while(validoAgressividade){
                String agressividadeJogadorLeitura = leitura.nextLine().trim();
                if(contemLetras(agressividadeJogadorLeitura)){
                    System.out.println("A agressividade deve ser um número.
Introduza novamente: ");
                    continue;
                }
                else if(agressividadeJogadorLeitura.trim().isEmpty()){
                    System.out.println("A posição deve ser um número.
Introduza novamente: ");
                    continue;
                }
                if(agressividadeJogadorLeitura.trim().length()>=1 &&
agressividadeJogadorLeitura.trim().length()<=3){
                    int agressividadeJogadorInt =
Integer.parseInt(agressividadeJogadorLeitura.trim());
                        if(agressividadeJogador(agressividadeJogadorInt)){
                            validoAgressividade = false;

jogadorAAdicionar.setNivel_agressividade(agressividadeJogadorInt);

                            break;
                        }
                        else{
                            System.out.println("Introduza um nível de
agressividade válido entre 0 e 100: ");
                            continue;
                        }
                }
                else{
```

```
                System.out.println("Introduza um nível de agressividade
válido entre 0 e 100: ");
                continue;
            }
        }
        boolean validoOverall = true;
      System.out.println("Introduza o overall entre 1 e 99:");
      while(validoOverall ){
        String overallLeitura = leitura.nextLine().trim();
        if(contemLetras(overallLeitura)){
            System.out.println("A overall não pode conter letras. Introduza
novamente: ");
            continue;
        }
        else if(overallLeitura.trim().isEmpty()){
             System.out.println("Introduza um overall válido");
            continue;
        }
        int overall=0;
        if(overallLeitura.trim().length()==1 ||
overallLeitura.trim().length()==2){
            overall = Integer.parseInt(overallLeitura.trim());
            if(overall >= 1 && overall <= 99){
                validoOverall = false;
                jogadorAAdicionar.setOverall(overall);
                break;
            }
            else{
                System.out.println("Introduza um overall válido entre 1 anos
e 99: ");
                continue;
            }
        }
        else{
            System.out.println("Introduza um overall válido entre 1 anos e
99: ");
            continue;
```

```
            }

        }
            jogadores.add(jogadorAAdicionar);
//            System.out.println(jogadorAAdicionar);
//            System.out.println("Equipa: " + jogadorAAdicionar.getEquipa());
    }

    public void registarTreinadorPrincipal(List<TreinadorPrincipal>
treinadores){
        TreinadorPrincipal treinadorAAdicionar = new TreinadorPrincipal();
        System.out.println("Introduza o nome do treinador: ");
        boolean validoNome = true;

        while(validoNome){
            String nomeTreinador = leitura.nextLine().trim();
            if (percorrerTreinadoresPrincipal(treinadores, nomeTreinador)){
                System.out.println("Esse treinador já existe, introduza outro");
                continue;
            }
            else if(contemNumeros(nomeTreinador)){
                System.out.println("O nome do treinador não pode ter
números, introduza um nome válido");
            }
            else if(nomeTreinador.trim().isEmpty()){
                 System.out.println("Introduza um nome válido");
                continue;
            }
            else{
                validoNome = false;
                treinadorAAdicionar.setNome(nomeTreinador.trim());
                break;
            }
        }

        boolean validoIdade = true;
```

```
        System.out.println("Introduza a idade do treinador entre 21 anos e
100 anos: ");
        while(validoIdade){
            String idadeTreinadorLeitura = leitura.nextLine().trim();
            if(contemLetras(idadeTreinadorLeitura)){
                System.out.println("A idade não pode conter letras. Introduza
novamente: ");
                continue;
            }
            else if(idadeTreinadorLeitura.trim().isEmpty()){
                 System.out.println("Introduza uma idade válida");
                continue;
            }
            int idadeTreinador=0;
            if(idadeTreinadorLeitura.trim().length()==1 ||
idadeTreinadorLeitura.trim().length()==2 ||
idadeTreinadorLeitura.trim().length()==3){
                idadeTreinador =
Integer.parseInt(idadeTreinadorLeitura.trim());
                if(idadeTreinador >= 21 && idadeTreinador <= 100){
                    validoIdade = false;
                    treinadorAAdicionar.setIdade(idadeTreinador);
                    break;
                }
                else{
                    System.out.println("Introduza uma idade válida entre 21
anos e 100 anos: ");
                    continue;
                }
            }
            else{
                System.out.println("Introduza uma idade válida entre 21 anos
e 100 anos: ");
                continue;
            }

        }
```

```
        System.out.println("Introduza a/as especializacoes do treinador: ");
        boolean especializacoesTreinador=true;

        while(especializacoesTreinador){
            String especializacao = leitura.nextLine().trim();

if(percorrerTreinadoresEspecializacoes(treinadorAAdicionar,especializac
ao)){
                System.out.println("Introduza uma especializacao não
repetida: ");
                continue;
            }
            else if(especializacao.trim().isEmpty()){
                System.out.println("Introduza uma especializacao válida: ");
                continue;
            }
            else if(contemNumeros(especializacao)){
                System.out.println("As especializacoes não podem conter
números. Introduza novamente: ");
                continue;
            }

treinadorAAdicionar.ADDEspecializacoes(treinadorAAdicionar.GetEspeci
alizacoes(), especializacao.trim());

            System.out.println("Deseja inserir mais alguma
especialização?\n1. SIM\n2. NAO");
            while(true){
                String opcao = leitura.nextLine().trim();
                if(contemLetras(opcao)){
                    System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                    continue;
                }
                else if(opcao.trim().isEmpty()){
                    System.out.println("Introduza uma opção válida");
```

```
                continue;
            }

            int opcaoInt=0;
            if(opcao.trim().length()==1){
                opcaoInt = Integer.parseInt(opcao);
                if(opcaoInt == 1){
                    System.out.println("Especialização: ");
                    break;
                }
                else if(opcaoInt == 2){
                    especializacoesTreinador=false;
                    break;
                }
                else{
                    System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                    continue;
                }
            }
            else{
                System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                continue;
            }

        }
    }
    System.out.println("Introduza a/as taticas do treinador respeitando
este formato XYZ: ");
    boolean taticasTreinador=true;
    while(taticasTreinador){
        String taticas = leitura.nextLine().trim();
        String taticaAAdicionar=adicionarSeparacaoTatica(taticas);

if(percorrerTreinadoresTaticas(treinadorAAdicionar,taticaAAdicionar)){
            System.out.println("Introduza uma tatica não repetida: ");
```

```
            continue;
        }
        else if(contemLetras(taticas)){
            System.out.println("Introduza uma tatica válida");
            continue;
        }
        else if(taticas.trim().isEmpty()){
            System.out.println("Introduza uma tatica válida");
            continue;
        }
        else{
            if(taticavalida(taticas)){

treinadorAAdicionar.ADDTaticaPref(treinadorAAdicionar.GetTaticaspref()
, taticaAAdicionar.trim());
            }
            else{
                System.out.println("Introduza uma tatica válida");
                continue;
            }

        }
        System.out.println("Deseja inserir mais alguma tática?\n1.
SIM\n2. NAO");
        while(true){
            String opcao = leitura.nextLine().trim();
            if(contemLetras(opcao)){
                System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                continue;
            }

            int opcaoInt=0;
            if(opcao.length()==1){
                opcaoInt = Integer.parseInt(opcao);
                if(opcaoInt == 1){
                    System.out.println("Tática: ");
```

```
                    break;
                }
                else if(opcaoInt == 2){
                    taticasTreinador=false;
                    break;
                }
                else{
                    System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                    continue;
                }
            }
            else{
                System.out.println("Escolha ou 1 ou 2. Introduza
novamente: ");
                continue;
            }

        }
    }
    boolean validoOverall = true;
    System.out.println("Introduza o overall entre 1 e 99:");
    while(validoOverall ){
        String overallLeitura = leitura.nextLine().trim();
        if(contemLetras(overallLeitura)){
            System.out.println("A overall não pode conter letras. Introduza
novamente: ");
            continue;
        }
        else if(overallLeitura.trim().isEmpty()){
             System.out.println("Introduza um overall válido");
            continue;
        }
        int overall=0;
        if(overallLeitura.trim().length()==1 ||
overallLeitura.trim().length()==2){
            overall = Integer.parseInt(overallLeitura.trim());
```

```
            if(overall >= 1 && overall <= 99){
                validoOverall = false;
                treinadorAAdicionar.setOverall(overall);
                break;
            }
            else{
                System.out.println("Introduza um overall válido entre 1 anos
e 99: ");
                continue;
            }
        }
        else{
            System.out.println("Introduza um overall válido entre 1 anos e
99: ");
            continue;
        }

    }
    treinadores.add(treinadorAAdicionar);
//      System.out.println(treinadorAAdicionar);
  }
  public void registarTreinadorAssistente(List<TreinadorAssistente>
treinadoresAssistente){
    TreinadorAssistente treinadorAAdicionar = new
TreinadorAssistente();
    System.out.println("Introduza o nome do treinador: ");
    boolean validoNome = true;

    while(validoNome){
        String nomeTreinador = leitura.nextLine().trim();
        if (percorrerTreinadoresAssistente(treinadoresAssistente,
nomeTreinador)){
            System.out.println("Esse treinador já existe, introduza outro");
            continue;
        }
        else if(contemNumeros(nomeTreinador)){
```

```
            System.out.println("O nome do treinador não pode ter
números, introduza um nome válido");
        }
        else if(nomeTreinador.trim().isEmpty()){
            System.out.println("Introduza um nome válido");
            continue;
        }
        else{
            validoNome = false;
            treinadorAAdicionar.setNome(nomeTreinador.trim());
            break;
        }
    }

    boolean validoIdade = true;
    System.out.println("Introduza a idade do treinador entre 21 anos e
100 anos: ");
    while(validoIdade){
        String idadeTreinadorLeitura = leitura.nextLine().trim();
        if(contemLetras(idadeTreinadorLeitura)){
            System.out.println("A idade não pode conter letras. Introduza
novamente: ");
            continue;
        }
        else if(idadeTreinadorLeitura.trim().isEmpty()){
            System.out.println("Introduza uma idade válida");
            continue;
        }
        int idadeTreinador=0;
        if(idadeTreinadorLeitura.trim().length()==1 ||
idadeTreinadorLeitura.trim().length()==2 ||
idadeTreinadorLeitura.trim().length()==3){
            idadeTreinador =
Integer.parseInt(idadeTreinadorLeitura.trim());
            if(idadeTreinador >= 21 && idadeTreinador <= 100){
                validoIdade = false;
                treinadorAAdicionar.setIdade(idadeTreinador);
```

```
                break;
            }
            else{
                System.out.println("Introduza uma idade válida entre 21
anos e 100 anos: ");
                continue;
            }
        }
        else{
            System.out.println("Introduza uma idade válida entre 21 anos
e 100 anos: ");
            continue;
        }

    }
    boolean validoPos=true;
    System.out.println("Introduza a posição preferida do treinador:\n" +
"1. GR\n2. DEFESA\n3. MEDIO\n4. ATACANTE");
        while(validoPos){
            String posicaoTreinadorLeitura = leitura.nextLine().trim();
            if(contemLetras(posicaoTreinadorLeitura)){
                System.out.println("A posição deve ser um número.
Introduza novamente: ");
                continue;
            }
            else if(posicaoTreinadorLeitura.trim().isEmpty()){
                System.out.println("Introduza uma opção válida");
                continue;
            }
            int posicaoTreinadorInt=0;
            if(posicaoTreinadorLeitura.length()==1){
                posicaoTreinadorInt =
Integer.parseInt(posicaoTreinadorLeitura);
                String posicaoTreinador =
posicaoAdvemOpcao(posicaoTreinadorInt);
                if(posicaoTreinador.equalsIgnoreCase("Essa posição não
existe")){
```

```
                    System.out.println("Introduza uma opção válida:\n1.
GR\n2. DEFESA\n3. MEDIO\n4. ATACANTE");
                }
                else{
                    validoPos = false;

treinadorAAdicionar.setPosicaopreferida(posicaoTreinador.trim());
                    break;
                }
            }
            else{
                System.out.println("Introduza uma opção válida:\n1. GR\n2.
DEFESA\n3. MEDIO\n4. ATACANTE");
                continue;
            }

        }
        boolean validoOverall = true;
        System.out.println("Introduza o overall entre 1 e 99:");
        while(validoOverall ){
            String overallLeitura = leitura.nextLine().trim();
            if(contemLetras(overallLeitura)){
                System.out.println("A overall não pode conter letras. Introduza
novamente: ");
                continue;
            }
            else if(overallLeitura.trim().isEmpty()){
                 System.out.println("Introduza um overall válido");
                continue;
            }
            int overall=0;
            if(overallLeitura.trim().length()==1 ||
overallLeitura.trim().length()==2){
                overall = Integer.parseInt(overallLeitura.trim());
                if(overall >= 1 && overall <= 99){
                    validoOverall = false;
                    treinadorAAdicionar.setOverall(overall);
```

```
                break;
            }
            else{
                System.out.println("Introduza um overall válido entre 1 anos
e 99: ");
                continue;
            }
        }
        else{
            System.out.println("Introduza um overall válido entre 1 anos e
99: ");
            continue;
        }

    }
        treinadoresAssistente.add(treinadorAAdicionar);

}
//So pode ter 1 guarda redes
public void registarEquipa(List<Liga> ligas,List<Jogador>
jogadores,List <Equipa> equipas,List <TreinadorPrincipal>
treinadores,List <TreinadorAssistente> treinadoresAssistente) {
    Equipa EquipaAAdicionar = new Equipa();
    int escolherTreinadorLeituraInt=0;
    int escolherGRLeituraInt=0;
    int valorLiga=-1;
    if(mostrarJogadoresPos("GR",jogadores).size()==0 ||
            mostrarJogadoresPos(jogadores).size()==0 ||
mostrarTreinadoresPrincipal(treinadores).size()==0
          ||
mostrarTreinadoresAssistente(treinadoresAssistente).size()==0){
        System.out.println("Não existe elementos suficientes para formar
uma equipa: ");
        return;
    }
    System.out.println("Introduza o nome da equipa: ");
    boolean validoNome=true;
```

```
        while(validoNome){
            String nomeEquipa = leitura.nextLine().trim();
            if(percorrerEquipas(equipas,nomeEquipa.trim())){
                System.out.println("Essa equipa já existe, introduza outra");
                continue;
            }
            else if(contemNumeros(nomeEquipa)){
                System.out.println("O nome da equipa não pode ter números,
introduza um nome válido");
                continue;
            }
            else if(nomeEquipa.trim().isEmpty()){
                System.out.println("Introduza um nome de equipa válido");
                continue;
            }
            else{
                validoNome=false;
                EquipaAAdicionar.setNome(nomeEquipa);
                break;

            }

        }

        boolean validoLiga = true;
        int percorrerLigasInt=0;
        System.out.println("Introduza a liga da equipa: ");
        String copialigaLeitura="";
        while(validoLiga){
            String ligaLeitura = leitura.nextLine().trim();
            copialigaLeitura=ligaLeitura;
            if(contemNumeros(ligaLeitura)){
                System.out.println("A equipa não pode conter números.
Introduza novamente: ");
                continue;
            }
            else if(ligaLeitura.trim().isEmpty()){
```

```
                System.out.println("Introduza uma liga válida");
                continue;
            }
            else{
              validoLiga=false;

              for(int z=0;z<ligas.size();z++){
                 if(ligas.get(z).getNome().equalsIgnoreCase(ligaLeitura)){

if(ligas.get(z).getjornada()<ligas.get(z).getequipas().size()+1){
                     EquipaAAdicionar.setLiga(ligas.get(z).getNome());
                     valorLiga=z;
                  }
                  else{
                    System.out.println("Não é possível adicionar uma equipa
depois de ter começado a segunda volta");
                    return;
                  }


                 }
              }
              break;
            }
        }
        boolean validoTreinadorPrincipal = true;
        int posTreinadorPrincipalInt=-1;
        System.out.println("Escolha um treinador principal: ");
        while(validoTreinadorPrincipal){
           for(int i=0;i<mostrarTreinadoresPrincipal(treinadores).size();i++){
              System.out.println(i + 1 + ". " +
mostrarTreinadoresPrincipal(treinadores).get(i).getNome());
           }
           String treinadorLeitura = leitura.nextLine().trim();


           if(contemLetras(treinadorLeitura)){
```

```
            System.out.println("Introduza uma opcao válida,sem letras: ");
            continue;
        }
        else if(treinadorLeitura.trim().isEmpty()){
             System.out.println("Introduza uma opcao válida");
            continue;
        }
        else if(mostrarTreinadoresPrincipal(treinadores).size()>0){
          posTreinadorPrincipalInt = Integer.parseInt(treinadorLeitura);
          posTreinadorPrincipalInt--;
          validoTreinadorPrincipal=false;

EquipaAAdicionar.setTreinadorPrincipal(mostrarTreinadoresPrincipal(trei
nadores).get(posTreinadorPrincipalInt));
          break;
        }
      }
      boolean validoTreinadorAssistente = true;
      int posTreinadorAssistenteInt=-1;
      System.out.println("Escolha um treinador assistente: ");
      while(validoTreinadorAssistente){
        for(int
i=0;i<mostrarTreinadoresAssistente(treinadoresAssistente).size();i++){
            System.out.println(i + 1 + ". " +
mostrarTreinadoresAssistente(treinadoresAssistente).get(i).getNome());
        }
        String treinadorLeituraAssistente = leitura.nextLine().trim();


        if(contemLetras(treinadorLeituraAssistente)){
            System.out.println("Introduza uma opcao válida,sem letras: ");
            continue;
        }
        else if(treinadorLeituraAssistente.trim().isEmpty()){
             System.out.println("Introduza uma opcao válida");
            continue;
        }
```

```
            else
if(mostrarTreinadoresAssistente(treinadoresAssistente).size()>0){
              posTreinadorAssistenteInt =
Integer.parseInt(treinadorLeituraAssistente);
              posTreinadorAssistenteInt--;
              validoTreinadorAssistente=false;

EquipaAAdicionar.setTreinadorAssistente(mostrarTreinadoresAssistente
(treinadoresAssistente).get(posTreinadorAssistenteInt));
              break;
            }
          }
          boolean validoGR = true;
          int posJogadorInt=0;
          System.out.println("Escolha um treinador: ");

          while(validoGR){
            for(int i=0;i<mostrarJogadoresPos("GR",jogadores).size();i++){
              System.out.println(i + 1 + ". " +
mostrarJogadoresPos("GR",jogadores).get(i).getNome() + "-" +

mostrarJogadoresPos("GR",jogadores).get(i).getPosJogador());
            }
            String gkLeitura = leitura.nextLine().trim();


            if(contemLetras(gkLeitura)){
              System.out.println("Introduza uma opcao válida,sem letras: ");
              continue;
            }
            else if(gkLeitura.trim().isEmpty()){
               System.out.println("Introduza uma opcao válida");
              continue;
            }
            else if(mostrarJogadoresPos("GR",jogadores).size()>0){
              posJogadorInt = Integer.parseInt(gkLeitura);
              posJogadorInt--;
```

```
            validoGR=false;

EquipaAAdicionar.addJogadores(jogadores,mostrarJogadoresPos("GR",
jogadores).get(posJogadorInt));
            break;
          }
        }
        boolean validoRestoJogadores = true;
        int posRestoJogadorInt=0;
        List<Integer> posJogadorAAdicionarInt= new ArrayList();
        System.out.println("Escolha os restantes jogadores: ");

while (validoRestoJogadores) {
   for (int i = 0; i < mostrarJogadoresPos(jogadores).size(); i++) {
   System.out.println(i + 1 + ". " +
mostrarJogadoresPos(jogadores).get(i).getNome() + "-" +
mostrarJogadoresPos(jogadores).get(i).getPosJogador());
   }
   String posLeitura = leitura.nextLine().trim();

   if (contemLetras(posLeitura) || posLeitura.isEmpty()) {
      System.out.println("Introduza uma opção válida, sem letras: ");
      continue;
   }

   posRestoJogadorInt = Integer.parseInt(posLeitura) - 1;

   if (posRestoJogadorInt < 0 || posRestoJogadorInt >=
mostrarJogadoresPos(jogadores).size()) {
      System.out.println("Introduza uma opção válida");
      continue;
   }

   posJogadorAAdicionarInt.add(posRestoJogadorInt);
   EquipaAAdicionar.addJogadores(jogadores,
mostrarJogadoresPos(jogadores).get(posRestoJogadorInt));
```

```
    if (EquipaAAdicionar.getJogadoresLista().size() == 11) {
        validoRestoJogadores = false;
        equipas.add(EquipaAAdicionar);
        break;
    }




}
    if(valorLiga!=-1){
        ligas.get(valorLiga).addequipa(EquipaAAdicionar);
    }
    if(!percorrerLigas(ligas,copialigaLeitura)){
       Liga novaLiga=new Liga(copialigaLeitura.trim());
       EquipaAAdicionar.setLiga(novaLiga.getNome());
       novaLiga.addequipa(EquipaAAdicionar);
       ligas.add(novaLiga);

        }
//System.out.println(ligas.get(valorLiga));
//System.out.println(ligas);
}

    public Jogador removerJogador(List<Jogador> lista, int indice) {
    Jogador jogadorRemovido = lista.remove(indice);
    return jogadorRemovido;
}
    public boolean percorrerArbitros(List<Arbitro> arbitros,String
nomeArbitro){
        for(int i=0;i<arbitros.size();i++){
            if(arbitros.get(i).getNome().equalsIgnoreCase(nomeArbitro)){
                return true;
            }
        }
        return false;
    }
```

```
    public void registarArbitro(List<Arbitro> arbitros) {
        Arbitro arbitroAAdicionar = new Arbitro();
        System.out.println("Introduza o nome do arbítro: ");
        boolean validoNome=true;
        while(validoNome){
            String nomeArbitro = leitura.nextLine().trim();
            if(percorrerArbitros(arbitros,nomeArbitro.trim())){
                System.out.println("Já existe um arbítro com esse nome,
introduza outro");
                continue;
            }
            else if(contemNumeros(nomeArbitro)){
                System.out.println("O nome do arbítro não pode ter números,
introduza um nome válido");
                continue;
            }
            else if(nomeArbitro.trim().isEmpty()){
                System.out.println("Introduza um nome válido");
                continue;
            }
            else{
                validoNome=false;
                arbitroAAdicionar.setNome(nomeArbitro.trim());
                break;
            }
        }
        boolean validoIdade = true;
        System.out.println("Introduza a idade do arbítro entre 16 anos e 50
anos: ");
        while(validoIdade){
            String idadeArbitroLeitura = leitura.nextLine().trim();
            if(contemLetras(idadeArbitroLeitura)){
                System.out.println("A idade não pode conter letras. Introduza
novamente: ");
                continue;
            }
            else if(idadeArbitroLeitura.trim().isEmpty()){
```

```
                System.out.println("Introduza uma idade válida");
                continue;
            }
            int idadeArbitro=0;
            if(idadeArbitroLeitura.trim().length()==1 ||
idadeArbitroLeitura.trim().length()==2){
                idadeArbitro = Integer.parseInt(idadeArbitroLeitura.trim());
                if(idadeArbitro >= 16 && idadeArbitro <= 50){
                    validoIdade = false;
                    arbitroAAdicionar.setIdade(idadeArbitro);
                    break;
                }
                else{
                    System.out.println("Introduza uma idade válida entre 16
anos e 50 anos: ");
                    continue;
                }
            }
            else{
                System.out.println("Introduza uma idade válida entre 16 anos
e 50 anos: ");
                continue;
            }

        }
        boolean validoanosExperienciaArbitro = true;
        System.out.println("Introduza os anos de experiencia entre 0 anos
e 30 anos: ");
        while(validoanosExperienciaArbitro){
            String anosExperienciaArbitroLeitura = leitura.nextLine().trim();
            if(contemLetras(anosExperienciaArbitroLeitura)){
                System.out.println("Anos de experiencia não pode conter
letras. Introduza novamente: ");
                continue;
            }
            else if(anosExperienciaArbitroLeitura.trim().isEmpty()){
                System.out.println("Introduza anos de experiencia válidos");
```

```
                continue;
            }
            int anosExperienciaArbitro=0;
            if(anosExperienciaArbitroLeitura.trim().length()==1 ||
anosExperienciaArbitroLeitura.trim().length()==2){
                anosExperienciaArbitro =
Integer.parseInt(anosExperienciaArbitroLeitura.trim());
                if(anosExperienciaArbitro >= 0 && anosExperienciaArbitro <=
30){
                    validoIdade = false;
                    arbitroAAdicionar.setIdade(anosExperienciaArbitro);
                    break;
                }
                else{
                    System.out.println("Introduza os anos de experiencia entre
0 anos e 30 anos: ");
                    continue;
                }
            }
            else{
                System.out.println("Introduza os anos de experiencia entre 0
anos e 30 anos: ");
                continue;
            }
    }
}
    public boolean percorrerLigas(List<Liga> ligas,String nomeLiga){
        for(int i=0;i<ligas.size();i++){
            if(ligas.get(i).getNome().equalsIgnoreCase(nomeLiga)){
                return true;
            }
        }
        return false;
    }

    public int percorrerLigasInt(List<Liga> ligas,String nomeLiga){
        for(int i=0;i<ligas.size();i++){
```

```
            if(ligas.get(i).getNome().equalsIgnoreCase(nomeLiga)){
                return i;
            }
        }
        return -1;
    }
    public List<Equipa> equipaLigasDisp(List<Equipa> equipas){
        List<Equipa> equipasQueNaoTemLiga=new ArrayList();
        for(int i=0;i<equipas.size();i++){
            if(equipas.get(i).getLiga().equals("")){
                equipasQueNaoTemLiga.add(equipas.get(i));
            }
        }
        return equipasQueNaoTemLiga;
    }
    public void voltarAoMenu(List<Liga> ligas, List<Equipa>
equipas,List<TreinadorPrincipal> treinadoresPrincipal,List<Jogador>
jogadores,List<Arbitro> arbitros,List<TreinadorAssistente>
treinadoresAssistente) {
        Menu menu = new Menu();
        while(true){
            menu.opcoes();
            menu.escolha(ligas, equipas, treinadoresPrincipal, jogadores,
arbitros, treinadoresAssistente, this);
        }
    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;
```

```
/**
 *
 * @author Renato
 */
public class Treinador extends Pessoa {
    private String equipaAtreinar="";

    public String getEquipaAtreinar() {
        return equipaAtreinar;
    }

    public void setEquipaAtreinar(String equipaAtreinar) {
        this.equipaAtreinar = equipaAtreinar;
    }


}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;

/**
 *
 * @author Renato
 */
public class TreinadorAssistente extends Treinador {
    private String posicaopreferida = "";

    public String getPosicaopreferida() {
        return posicaopreferida;
    }
```

```
    public void setPosicaopreferida(String posicaopreferida) {
        this.posicaopreferida = posicaopreferida;
    }

    public String toString(){
        return "Treinador Assistente: " + this.nome + "\nIdade: " + this.idade
+ "\nEquipa atual: "+this.getEquipaAtreinar()+"\nPosicao preferida: " +
this.posicaopreferida+
                "\nOverall: " + this.overall;



    }
}

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.poo.projetofase2;
import java.util.*;
/**
 *
 * @author Renato
 */
public class TreinadorPrincipal extends Treinador {
    private List<String> especializacoes = new ArrayList<String>();
    private List<String> taticaspref = new ArrayList<String>();
    private int moral = 0;

    public TreinadorPrincipal(){

    }
    public TreinadorPrincipal(String nome, int idade, int overall){
```

```
        this.nome = nome;
        this.idade = idade;
        this.overall = overall;
    }
    public int getMoral() {
        return this.moral;
    }

    public void setMoral(int moral) {
        this.moral = moral;
    }

    public List<String> GetEspecializacoes(){
        return especializacoes;
    }

    public List<String> GetTaticaspref(){
        return taticaspref;
    }


    /*Adiciona especializacao e retorna todas as especializacoes*/
    public List<String> ADDEspecializacoes_LISTA(List<String>
especializacoes,String especializacao){
        especializacoes.add(especializacao);

        return especializacoes;
    }

    /*Adiciona especializacao*/
    public void ADDEspecializacoes(List<String> especializacoes,String
especializacao){
        especializacoes.add(especializacao);
    }

    public List<String> ADDTaticaPref_LISTA(List<String>
taticaspref,String tatica){
```

```
        taticaspref.add(tatica);

        return taticaspref;
    }

    public void ADDTaticaPref(List<String> taticaspref,String tatica){
        taticaspref.add(tatica);
    }


   public String toString(){
        return "Treinador: " + this.nome + "\nIdade: " + this.idade +
"\nEquipa atual: "+this.getEquipaAtreinar()+"\nEspecializacoes: " +
especializacoes +
                "\nTaticas preferidas: " + taticaspref +"\nOverall: " +
this.overall;

    }
}
```