Universidade do Minho
Escola de Ciências

Relatório do Projeto de Programação Orientada aos Objetos

LCC

2º Semestre

2020/2021

Grupo 63

Ana Beatriz Silva (91678)

Paulo Ricardo Costa (87986)

Rui Miguel Batista (87989)

Índice:

1.Introdução

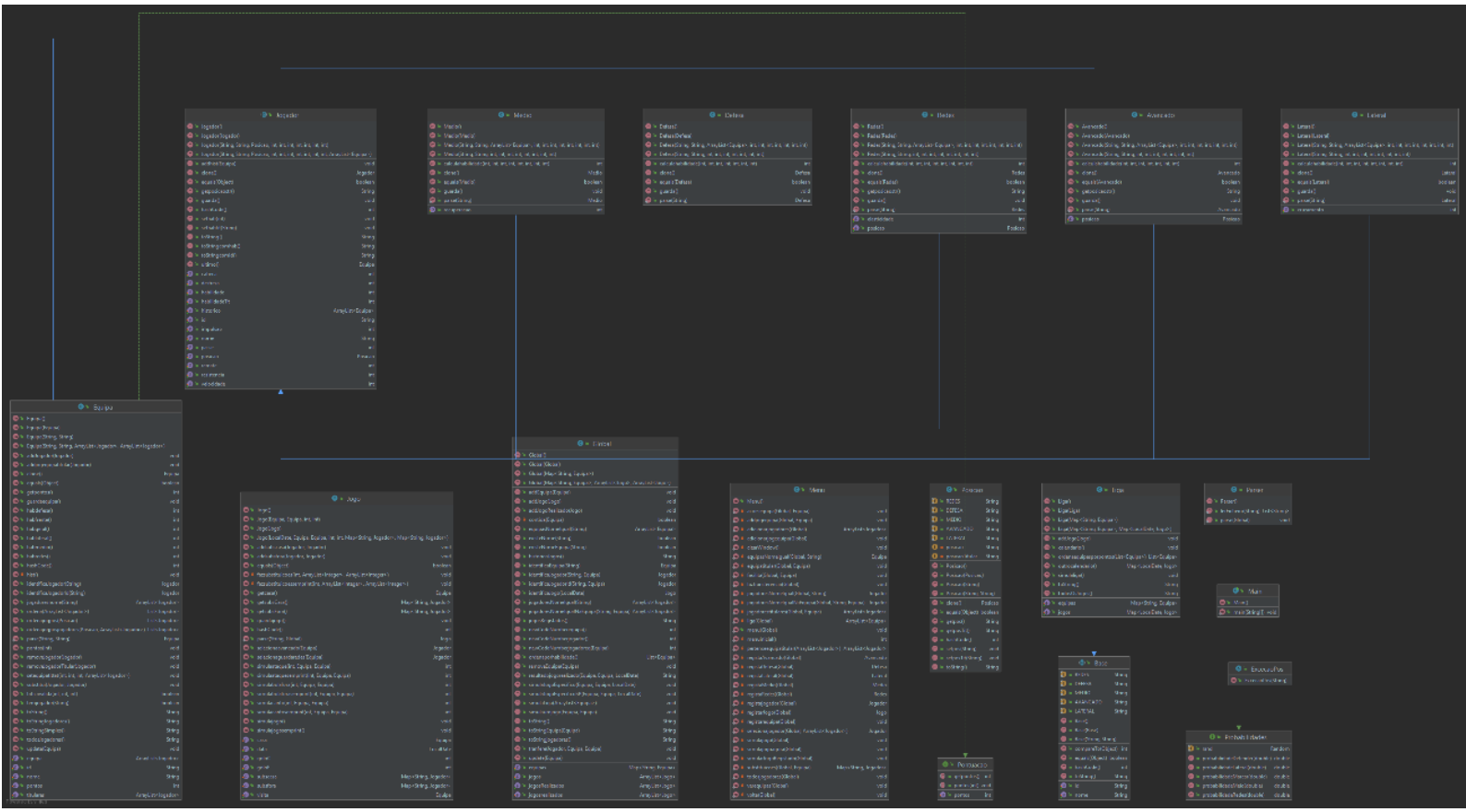
Este relatório aborda o projeto realizado pelo grupo nº63 na implementação de uma aplicação que cria um sistema de gestão e simulação de equipas de um jogo de futebol.

Nesta aplicação vai haver a possibilidade de criar o número de jogadores que o usuário quiser, e colocá-los em equipas. Cada jogador vai ser de um determinado tipo: guarda-redes, defesa, médio, avançado ou lateral.

Por sua vez, cada um vai possuir um determinado conjunto de características que incluem a velocidade, resistência, destreza, impulsão, jogo de cabeça, remate e capacidade de passe. Para alem destas, os laterais vão ter também de ter em consideração a sua capacidade para realizar cruzamentos, os guarda-redes vão ter a elasticidade, enquanto os médios vão ter a capacidade de recuperar a bola. De acordo com estas qualidades, cada jogador vai obter um valor que demonstra o seu nível de habilidades. Quanto maior foi esse valor, melhor vai ser o seu desempenho.

A habilidade global de cada equipa é calculada a partir das habilidades individuais de cada jogador, e a equipa com um maior valor de habilidade, vai ter uma maior probabilidade de ganhar o jogo.

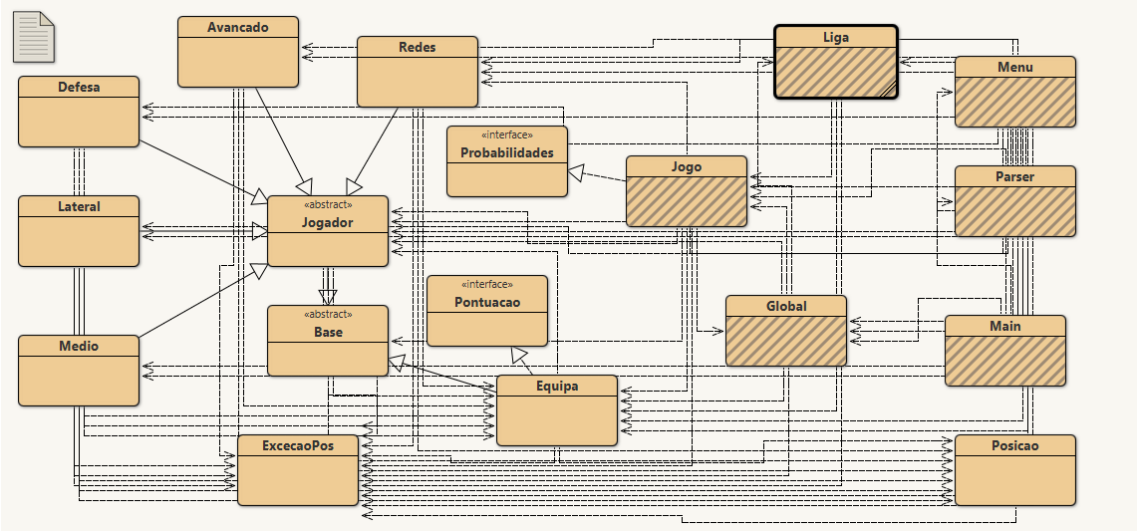
2. Arquitetura de Classes



Devido a se tratar de um diagrama bastante extenso, colocamos de seguida o link para visualização da imagem, para que se possa efetuar uma visualização mais concreta e onde possamos dar zoom, nas diferentes classes e nas que realmente pretendermos visualizar.

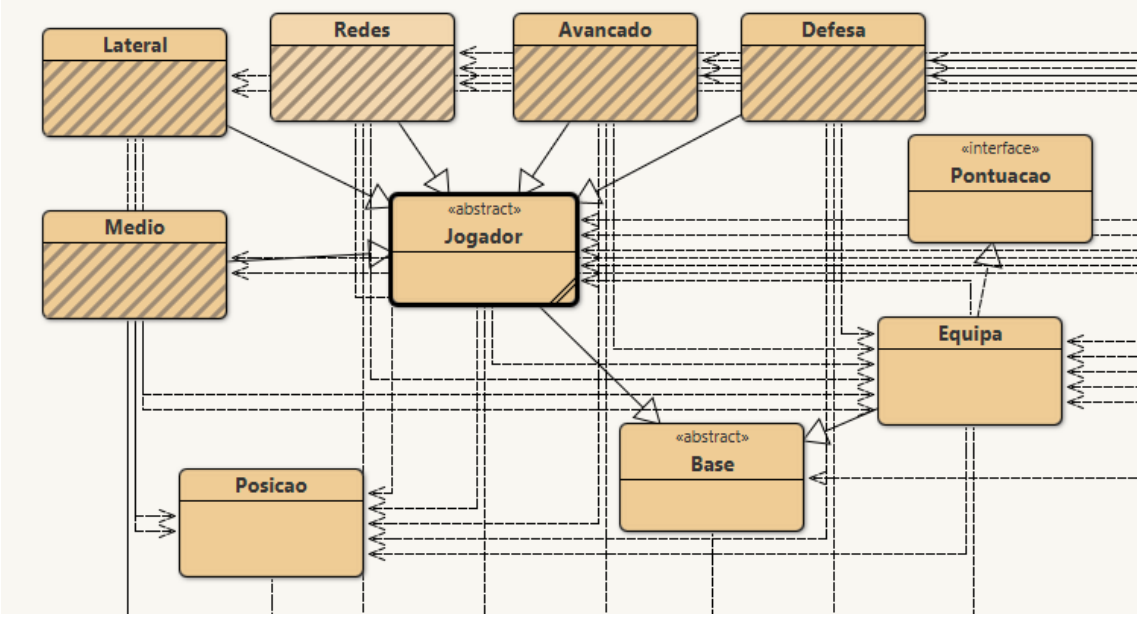
[Imagem Diagrama de Classes.]

A baixo apresenta-se um diagrama de classes simplificado, que não inclui métodos ou variáveis e vamos proceder por explicar um pouco cada uma das classes e falar das suas funções no programa.



2.1 Base e as classes e heranças a estas relacionadas

A baixo apresenta-se um diagrama simplificado com algumas classes presentes no projeto. Neste tópico vamos abordar cada uma delas e ver qual o seu propósito na aplicação.



2.1.1 Base

A classe base pode ser considerada a classe "pai" das classes Equipa e Jogador. Todas as classes que vamos ver a seguir podem ser consideradas subclasses da Base.

Variáveis de instância:

-static final String REDES = "Guarda-Redes"

-static final String DEFESA = "Defesa"

-static final String MEDIO = "Medio"

-static final String AVANCADO= "Avancado"

-static final String LATERAL = "Lateral"

-String nome

-String id

Métodos de classe:

-Base()

-Base(String nome, String id)

-Base (Geral a)

-boolean equals(Object o)

-int hashCode()

-String toString()

-int compareTo(Object o)

2.1.2 Jogador

A classe Jogador vai herdar as variáveis da classe Geral. Esta classe trata-se de uma classe abstrata.

Variáveis de instância:

-Posicao posicao

-int velocidade [(1)]

-int resistencia [(1)]

-int destreza [(1)]

-int impulsao [(1)]

-int cabeca [(1)]

-int remate [(1)]

-int passe [(1)]

-int habilidade [(2)]

-int habilidadeTit [(3)]

- ArrayList<Equipa> histórico [(4)]

Métodos:

-Jogador()

-Jogador(String id, String nome, Posicao pos, int v, int r, int d, int i, int c, int remate, int passe, ArrayList<Equipa>a) [(5)]

-Jogador(String id, String nome, Posicao pos, int v, int r, int d, int l, int c, int remate, int passe) [(6)]

-Jogador (Jogador e)

-Posicao getPosicao()

-String getposicaostr()

-Equipa ultimo() [(7)]

-addhist (Equipa a) [(8)]

-boolean equals(Object o)

-hashCode()

-String toString() [(9)]

-String toStringComid() [(9)]

-String toStringcomhab() [(9)]

-void sethabtit(String pos) [(10)]

-void save()

-Jogador clone()

(1) Estas variáveis de instância vão estabelecer o valor de habilidade que cada vai jogador possuir para cada um dos fatores;

(2) Vai fornecer o valor da habilidade geral de cada jogador, juntando os valores que são dados pelas variáveis mencionadas em (1) numa fórmula que vai determinar esse valor;

(3)Refere-se há habilidade que cada jogador possui numa certa posição, caso esta seja a sua posição preferida, então esse valor, quando joga na equipa titular, vai ser igual ao da habilidade geral(2).

(4)A lista Equipa vai estabelecer a equipa em que cada jogador é colocado, sendo a última posição no seu histórico a equipa a que este pertence.

(5)Jogador com histórico;

(6)Jogador sem histórico;

(7) Vê a última equipa do histórico de um jogador e essa é a sua equipa atual;

(8)Adiciona uma equipa ao histórico (é usado na classe equipa);

(9) Temos diferentes to string, isto para diferentes ocasiões; ver jogador e ver jogador em específico;

(10) Método que vê se a habilidade do jogador numa certa posição; caso esta seja a sua posição preferível, a habilidade não muda quando este for titular, caso contrário haverá penalizações e o valor vai diminuir; por exemplo, casa um lateral seja colocado como defesa titular, a sua habilidade nessa posição será a sua habilidade geral menos 0.22 dessa mesma habilidade. (habilidade titular = habilidade – 0.22*habilidade).

2.1.3 Avançado

Os avançados são, no futebol, os jogadores em que se coloca as maiores expetativas durante os jogos, pois são estes que tem a responsabilidade de marcar golos.

Variáveis de instância:

Como a classe Avançado é uma das classes filho de Base, vai imediatamente herdar as suas variáveis.

Métodos de classe:

-Avancado()

-Avancado(String id, String nome, ArrayList<Equipa> a, int v, int r, int d, int i, int c, int remate, int p)[(1)]

-Avancado(String id, String nome, int v, int r, int d, int i, int c, int remate, int p)[(2)]

-Avancado(Avancado e)

-Posicao getPosicao()

-calculahabilidade(int v, int r, int d, int i, int c, int remate, int p) [(3)]

-boolean equals(Avancado e)

-void guarda()[(4)]

-Avancado clone()

-static Avancado parse(String input)

[(1)]Avançado com histórico;

[(2)]Avançado sem histórico;

[(3)]Método que é usado para calcular a habilidade geral de entre todas as habilidades do jogador;

[(4)]Método que é usado para escrever no ficheiro. Criamos o BufferedWriter com esse propósito, e este tem como argumento o caminho para o ficheiro e true. Usamos ainda o escritor.write para escrever no ficheiro e escritor.flush para o atualizar.

2.1.4 Médio

Os médios são responsáveis por fazer a ligação entre a defesa e o ataque da equipa, e atuam tanto nas jogadas ofensivas como nas defensivas.

Variáveis de instância:

Como a classe Médio é uma das classes filho de Base, vai, tal como a classe Avançado, herdar as suas variáveis. Vai ainda ter a variável int recuperação.

Métodos de classe:

- Medio()
- Medio(String id,String nome, ArrayList<Equipa> a ,int v, int r, int d, int i, int c, int remate, int p, int recuperacao) (1)
- Medio(String id,String nome, int v, int r, int d, int i, int c, int remate, int p, int recuperacao) (2)
- Medio(Medio e)
- int calculahabilidade(int v, int r, int d, int i, int c, int remate, int p ,int recuperacao) (3)
- boolean equals(Medio e)
- Medio clone()
- void guarda() (4)
- static Medio parse(String input)

(1) Médio com histórico;

(2) Médio sem histórico;

(3) Método que é usado para calcular a habilidade geral de entre todas as habilidades do jogador;

(4)Método que é usado para escrever no ficheiro. Criamos o BufferedWriter com esse propósito, e este tem como argumento o caminho para o ficheiro e true. Usamos ainda o escritor.write para escrever no ficheiro e escritor.flush para o atualizar.

2.1.5 Defesa

Os defesas estão normalmente posicionados entre a linha de meio-campo e a baliza, e tem como principal função, impedir os atacantes da equipa oposta de marcar.

Variáveis de instância:

Como a classe Defesa é uma das classes filho de Base, vai imediatamente herdar as suas variáveis.

Métodos de classe:

-Defesa()

- Defesa(String id,String nome, ArrayList<Equipa> a ,int v, int r, int d, int i, int c, int remate, int p)[1]

- Defesa(String id,String nome, int v, int r, int d, int i, int c, int remate, int p)[2]

- Defesa(Defesa e)

- int calculaHabilidade(int v, int r, int d, int i, int c, int remate, int p)[3]

- boolean equals(Defesa e)

- Defesa clone()

- void guarda ()[4]

- static Defesa parse(String input)

[1]Defesa com histórico;

[2]Defesa sem histórico;

[3]Método que é usado para calcular a habilidade geral de entre todas as habilidades do jogador;

[4]Método que é usado para escrever no ficheiro. Criamos o BufferedWriter com esse propósito, e este tem como argumento o caminho para o ficheiro e true. Usamos ainda o escritor.write para escrever no ficheiro e escritor.flush para o atualizar.

2.1.6 Lateral

Os laterais, como indica o nome, ocupam as laterais do campo e ajudam o guarda-redes a proteger a baliza. São também responsáveis por repor a bola quando ela sai do campo pelas linhas laterais.

Variáveis de instância:

Como a classe Lateral é uma das classes filho de Base, vai imediatamente herdar as suas variáveis. Vai ter ainda a variável int cruzamento.

Métodos de classe:

- Lateral()

- Lateral(String id,String nome, ArrayList<Equipa> a ,int v, int r, int d, int i, int c, int remate, int p, int cruzamento)[(1)]

- Lateral(String id,String nome, int v, int r, int d, int i, int c, int remate, int p, int cruzamento)[(2)]

- Lateral(Lateral e)

- int calculaHabilidade(int v, int r, int d, int i, int c, int remate, int p ,int cruza)[(3)]

- boolean equals(Lateral e)

- Lateral clone()

- void guarda()[(4)]

- static Lateral parse(String input)

[(1)]Lateral com histórico;

[(2)]Lateral sem histórico;

[(3)]Método que é usado para calcular a habilidade geral de entre todas as habilidades do jogador;

[(4)]Método que é usado para escrever no ficheiro. Criamos o BufferedWriter com esse propósito, e este tem como argumento o caminho para o ficheiro e true. Usamos ainda o escritor.write para escrever no ficheiro e escritor.flush para o atualizar.

2.1.7 Guarda-Redes

Os guarda-redes são responsáveis por impedir que a bola entre na sua baliza.

Variáveis de instância:

Como a classe Redes é uma das classes filho de Base, vai imediatamente herdar as suas variáveis. Vai ter ainda a variável int elasticidade.

Métodos de classe:

- Redes()

- Redes(String id,String nome, ArrayList<Equipa> a ,int v, int r, int d, int i, int c, int remate, int p, int e)[1]

- Redes(String id,String nome, int v, int r, int d, int i, int c, int remate, int p, int e)[2]

- Redes(Redes e)

- int calculaHabilidade(int v, int r, int d, int i, int c, int remate, int p ,int e)[3]

- boolean equals(Redes e)

- Redes clone()

- void guarda()[4]

- static Redes parse(String input)

[1]Redes com histórico;

[2]Redes sem histórico;

[3]Método que é usado para calcular a habilidade geral de entre todas as habilidades do jogador;

[4]Método que é usado para escrever no ficheiro. Criamos o BufferedWriter com esse propósito, e este tem como argumento o caminho para o ficheiro e true. Usamos ainda o escritor.write para escrever no ficheiro e escritor.flush para o atualizar.

2.1.8 Posição

A classe Posição é utilizada na classe Jogador e tem como objetivo determinar em que posição (avançado, lateral, médio, defesa, guarda-redes) cada jogador toma no campo, e se se trata de um jogador titular ou não.

Variáveis de instância:

-String REDES = "Guarda-Redes"

-String DEFESA = "Defesa"

-String MEDIO = "Medio"

-String AVANCADO = "Avancado"

-String LATERAL = "Lateral"

-String posicao

-String posicaoTitular

Métodos de classe:

-Posicao()

-Posicao(String pos)

-Posicao(String pos, String titu)

-Posicao(Posicao pos)

-void setPosicao(String pos)[(1)]

-void setPosicaoTitular(String pos)[(2)]

-String toString()

-boolean equals (Object o)

-int hashCode()

-Posicao clone()

[(1)]Método que descobre qual a posição favorita do jogador;

[(2)]Método que descobre qual a posição titular em que o jogador vai tomar.

2.1.9 Equipa

Nesta classe vamos ver todos os métodos que fazem parte da construção de uma equipa.

Variáveis de instância:

-ArrayList<Jogador> equipa

-ArrayList<Jogador> titulares

-int pontos

Métodos de classe:

-Equipa()

-Equipa(String nome, String id)

-Equipa(String nome, String id, ArrayList<Jogador> a, ArrayList<Jogador> b)

-Equipa(Equipa b)[1]

-void addJogador(Jogador a)[2]

-void removeJogador(Jogador a)[3]

-void removeJogadorTitular(Jogador a)[4]

-void addjogequipatitular(Jogador a)[5]

-void substitui(Jogador entra, Jogador sai)[6]

-int habfrente()[7]

-int hablateral()[7]

-int habmedio()[7]

-int habdefesa()[7]

-int habredes()[7]

-int habgeral()

-boolean equals(Object o)

-int hashCode()

-String toStringJogadores()

-String todosJogadores()

-String toStringSimples()

-String toString()

-Equipa clone()

-void hist()[8]

-Jogador identificaJogadorID(String id)

-boolean temjogador(String nome)

-Jogador identificaJogador(String nome)

-ArrayList<Jogador> jogadoresnome(String nome)

-void update(Equipa a)[9]

-boolean taticaValida(int nrdefesas, int nrmedios, int nravancados)[10]

-List<Jogador> ordenajogpos(String pos)[11]

-List<Jogador> ordenajogposjogadores(String pos, ArrayList<Jogador> equipa)

-List<Jogador> ordena(ArrayList<Jogador> equipa)

-void setTaticaTitular(int nrdefesas, int nrmedios, int nravancados, ArrayList<Jogador> equipa)

-void pontos(int pontos)

-void guardaequipa()[12]

-static Equipa parse(String input, String id)

[1]Neste construtor não é necessário fazer um update ao histórico pois a equipa já vai incluir o histórico;

[2]Método que vai ser usado para adicionar um jogador à equipa, mas apenas será possível se este não estiver já nesta, pois não pode haver dois jogadores iguais na equipa;

[3]Método que remove um jogador da equipa;

[4] Método que remove um jogador da equipa titular;

[5]Método que adiciona um jogador à equipa titular, a não ser que, tal como em [2], este já esteja lá presente;

[6]Método que efetua uma substituição entre jogadores da equipa que são titulares e jogadores que não são titulares;

[7]Métodos que calculam a habilidade de cada posição da equipa;

[8]Método que faz o update do histórico das equipas em que um jogador já esteve;

[9]Método que faz o update de uma equipa;

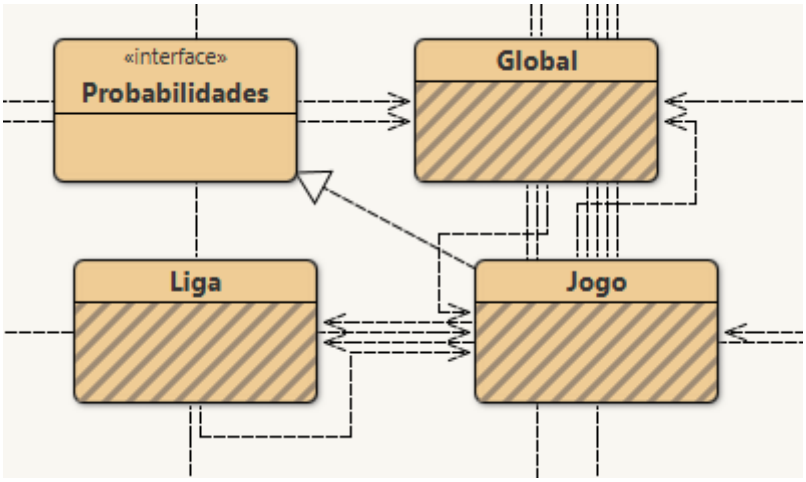
[10]Método que verifica se uma tática é válida ou não;

[11]Método que ordena os jogadores da equipa em função do valor de habilidade que estes possuem;

[12]Método que guarda o nome da equipa e atualiza-a, após isso, escreve o nome de cada jogador que pertence à equipa, volta a atualizar e no fim fecha-a.

2.2 Sistema de Simulação de Jogos

Agora vamos abordar o sistema que foi criado para simular jogos entre as diversas equipas criadas. Como podemos ver no excerto do diagrama, vamos falar de quatro classes diferentes, todas elas igualmente importantes para a aplicação.



2.2.1 Jogo

É nesta classe que são simulados os jogos entre duas equipas, desta forma, em vez de apresentarmos apenas o resultado de um jogo entre duas equipas, podemos ver o que acontece há medida que este está a decorrer. Para isso temos a data em que o jogo ocorre, as equipas, e número de golos que cada uma marca e também as substituições que vão ocorrer.

Variáveis de instância:

-LocalDate data

-Equipa casa

-Equipa visitante

-int goloC

-int goloF

-Map<String, Jogador> subscasa

-Map<String, Jogador> subsfora

Métodos de classe:

-Jogo()

-Jogo(Equipa a, Equipa b, int c, int f)

-Jogo(LocalDate data, Equipa a, Equipa b, int gc, int gf, Map<String, Jogador>subscasa, Map<String, Jogador> subsfora)

-Jogo(Jogo a)

-void addsubscasa(Jogador e, Jogador s)(1)

-void addsubsfora(Jogador e, Jogador s)(2)

-void fazsubstituicoes(int time,ArrayList<Integer>temposubscasa,ArrayList<Integer>temposubsfora)(3)

-void simulaJogo()(4)

- int simulaataque (int time, Equipa ataca, Equipa defende)(5)

-int simulacanto(int time, Equipa fazocanto, Equipa defende)(6)

- int simulabolafora(int time, Equipa ataca, Equipa defende)(7)

-Jogador selecionaguardaredes(Equipa e)(8)

-Jogador selecionaavancado(Equipa e)(9)

-boolean equals(Object o)

-int hashCode()

-static Jogo parse(String input, Geral a)

-void fazsubstituicoessemprint(int time,ArrayList<Integer>temposubscasa,ArrayList<Integer>temposubsfora)

-void simulajogosemprint()[10]

-int simulaataquesemprint (int time, Equipa ataca, Equipa defende)[11]

-int simulacantosemprint(int time, Equipa fazocanto, Equipa defende)[12]

-int simulabolaforasemprint(int time, Equipa ataca, Equipa defende)[13]

- void guardajogo()[14]

[1]Método que adiciona uma substituição na equipa que está a jogar em "casa";

[2]Método que adiciona uma substituição na equipa de fora;

[3]Método que recebe o tempo das substituições de casa e de fora (tempos aleatórios), e quando no ciclo, o tempo em que o jogo vai, é igual ao tempo de uma substituição, esta é realizada e retirada;

[4]Método que simula o jogo. Não são aceites substituições antes dos 5 minutos ou após os 85. A bola começa na posse da equipa de casa, e as habilidades dos médios são comparadas; a equipa cujos médios tem um maior valor de habilidades, realiza o primeiro ataque;

[5]Método que simula um ataque. Estes podem ser feitos pelo centro do campo ou pelas laterais. O inteiro que é retornado representa o tempo que o ataque demorou;

[6]Método que simula um ataque pela lateral. É idêntico ao [5];

[7]Método que simula o que acontece quando a bola sai fora do campo;

[8]Método que retorna o nome do guarda-redes que executou a defesa da jogada em que vamos;

[9]Método que retorna o nome do avançado que marcou um golo na jogada em que vamos;

[10]Método que vai imprimir/ escrever no ecrã aquilo que foi feito em [4]

[11] Método que vai imprimir/ escrever no ecrã aquilo que foi feito em [5]

[12] Método que vai imprimir/ escrever no ecrã aquilo que foi feito em [6]

[13] Método que vai imprimir/ escrever no ecrã aquilo que foi feito em [7]

[14] Método que vai ter todas as informações sobre o que aconteceu ao longo do jogo, como as substituições, data do jogo, nomes das equipas e as suas pontuações finais.

2.2.2 Probabilidades

Esta interface é utilizada na simulação de jogos, em particular, é usada pela classe Jogo, e calcula todas as probabilidades para os eventos que vão ocorrer durante o jogo.

Como se pode observar, todos os métodos deste interface são métodos default, pois estes permitem-nos adicionar novas funcionalidades às interfaces já existentes sem interromper as suas implementações anteriores.

Métodos de classe:

-double probabilidadeMarcar(double frente)[(1)]

-double probabilidadeDefender(double defesa)[(2)]

-double probabilidadeMeio(double medio)[(3)]

-double probabilidadeRedes(double redes)[(4)]

-double probabilidadeLateral(double lateral)[(5)]

[(1)]Método que vai calcular qual a probabilidade de os avançados conseguirem marcar golo; este é usado em int simulaAtaque e int simulaCanto;

[(2)]Método que calcula a probabilidade que há de os defesas conseguirem defender um ataque da equipa oposta; este é usado em int simulaAtaque e int simulaCanto;

[(3)]Método vai calcular qual a probabilidade que os médios de cada equipa tem de realizar o primeiro ataque do jogo; este é usado em void simulaJogo();

[(4)]Método calcula a probabilidade de o guarda-redes defender a baliza contra o ataque da equipa adversária; este é usado em int simulaAtaque;

[(5)]Método calcula a probabilidade de os jogadores laterais conseguirem fazer um cruzamento com sucesso; este é usado em int simulaCanto;

2.2.3 Global

A classe Global trata-se de uma classe que tal como o nome indica, é responsável por muitas tarefas. É nesta classe que vamos ter os métodos encarregues de adicionar e remover jogadores, realizar transferências de jogadores, entre outras coisas.

Variáveis de instância:

-private Map<String, Equipa> equipas

-private ArrayList<Jogo> jogos

-private ArrayList<Jogo> jogosrealizados

Métodos de classe:

- Global()

- Global (Map<String, Equipa> equipas)

- Global (Map<String, Equipa> equipas,ArrayList<Jogo> jogos,ArrayList<Jogo> jogosrealizados)

- Global (Global e)

- Map<String, Equipa> getEquipas()

- ArrayList<Jogo> getJogos()

- ArrayList<Jogo> getJogosRealizados()

- void addEquipa(Equipa e)(1)

- void addJogo(Jogo e)(2)

- void addJogoRealizado(Jogo e)(3)

- void removeEquipa(Equipa e)(4)

- void tranfere(Jogador jogador, Equipa sai, Equipa entra)(5)

- boolean contida (Equipa e)(6)

- void simulaumjogo(Equipa casa, Equipa visita)(7)

- int newCodeNumberequipa()(8)

- int newCodeNumberjogador()(9)

- String toString()

- String toStringEquipa(Equipa a)(10)

- String toStringJogadores()(11)

- String jogosRegistados()(12)

- String historicoJogos()(13)

- void simulaJogoEspecifico(Equipa casa, Equipa fora, LocalDate data)(14)

- void simulaJogoEspecificoSP(Equipa casa, Equipa fora, LocalDate data)

- String resultadojogorealizado(Equipa casa, Equipa fora, LocalDate data)[15]

- boolean existenomeEquipa(String nome)[16]

- boolean exiteNome(String nome)[17]

- Jogo identificaJogo(LocalDate data)[18]

- Equipa identificaEquipa(String nome)[19]

- Jogador identificaJogador(String nome, Equipa e)[20]

- Jogador identificaJogadorId(String id, Equipa e)[21]

- void update(Equipa a)[22]

- ArrayList<Equipa> equipasNomeIgual(String nome)[23]

- ArrayList<Jogador> jogadoresNomeIgual(String nome)[24]

-ArrayList<Jogador> jogadoresNomeIgualNaEquipa(String nome, Equipa e)[25]

-List<Equipa> ordenaporhabilidade()[26]

-void simulaChampions(ArrayList<Equipa> equipas)

[1]Método que adiciona uma equipa;

[2]Método que adiciona um jogo;

[3]Método que adiciona um jogo já realizado;

[4]Método que remove uma equipa;

[5]Método que realiza transferências de jogadores entre equipas;

[6]Método que verifica se uma equipa existe na lista das equipas;

[7]Método que simula um jogo entre duas equipas;

[8]Método que gera um novo código para uma equipa. Cada equipa é identificada por um código diferente, e não existem equipas com o mesmo id;

[9]Método que gera um novo código para um jogador. Tal como nas equipas, cada jogador é identificado por um código diferente, não havendo dois jogadores com o mesmo id;

[10]Método que nos dá os nomes das equipas registadas;

[11]Método que nos dá os nomes dos jogadores registados;

[12]Método que nos dá informações sobre os jogos registados, como a data em que o jogo vai decorrer e as equipas que vão jogar uma contra a outra;

[13]Método que nos dá o histórico dos jogos, ou seja, que nos diz que jogos já ocorreram;

[14]Método que vai simular um jogo;

[15]Método que nos dá o resultado de um jogo, dizendo quantos pontos cada equipa obteve;

[16]Método que verifica se o nome de uma equipa já existe ou não;

[17]Método que verifica se o nome de um jogador existe ou não;

[18]Método que verifica se um jogo já existe ou não;

[19]Método que identifica uma equipa pelo nome;

[20]Método que identifica um jogador numa equipa pelo seu nome;

[21]Método que identifica um jogador numa equipa pelo seu id;

[22]Método que faz um update das informações de uma equipa;

[23]Método que retorna todas as equipas que estão registadas com o mesmo nome

[24]Método que retorna todos os jogadores que estão registados com o mesmo nome;

[25]Método que devolve todos os jogadores que estão registados com o mesmo nome dentro de uma equipa;

[26]Método que ordena as equipas por valor de habilidade de forma decrescente;

2.2.4 Champions

Esta classe vai funcionar como uma espécie de torneio entre várias equipas, que se enfrentam umas às outras com o objetivo de serem coroadas com o título de melhor equipa.

Variáveis de instância:

-Map<String, Equipa> equipas

-Map<LocalDate, Jogo> jogos

Métodos de classe:

- Champions()
- Champions (Map<String, Equipa> equipas)
- Champions (Map<String, Equipa> equipas, Map<LocalDate, Jogo> jogos)
- Champions (Champions e)
- Map<String, Equipa> getEquipas()
- Map<LocalDate, Jogo> getJogos()
- void setEquipas (Map<String, Equipa> eq)[(1)]
- void setJogos(Map<LocalDate,Jogo> jogos)[(2)]
- void addJogo(Jogo j)[(3)]
- Map<LocalDate,Jogo> outrocalendario()
- void calendario()
- List<Equipa> ordenaequipasporpontos(List<Equipa> eq)[(4)]
- void simulaChampions()[(5)]
- String toString()
- String todosOsJogos()[(6)]

[(1)]Método que seleciona as equipas que vão participar no torneio;

[(2)]Método que marca as datas e locais dos jogos;

[(3)]Método que adiciona um jogo ao torneio;

[(4)]Método que ordena as equipas pelo número de pontos que estas obtiveram para descobrir qual está em primeiro lugar;

[(5)]Método que simula o torneio;

[(6)]Método que faz a lista de todos os jogos que vão decorrer ao longo da champions.

2.3 Menu

A classe Menu vai ter a função que o View teria num modelo MVC, ou seja, o menu é a classe responsável pela saída da representação de dados. Esta é também a classe responsável pela interação com a pessoa a usar a aplicação.

Métodos de classe:

-int menuinicial() [(1)]

-void registarequipa(Global a) [(2)]

-void verequipas(Global a) [(3)]

-void acoesequipa(Global a, Equipa equipa) [(4)]

-void facilita(Global a, Equipa equipa)

-Jogador selecionajogador(Global a, ArrayList<Jogador> jogs) [(5)]

-ArrayList<Jogador> adicionarjogadores(Global a) [(6)]

-Defesa registaDefesa(Global a) [(7)]

-Medio registaMedio(Global a) [(7)]

-Lateral registaLateral(Global a) [(7)]

-Avancado registaAvancado(Global a) [(7)]

-Redes registaRedes(Global a) [(7)]

-Jogador registajogador(Global a) [(8)]

-void todosjogadores(Global a) [(9)]

-void clearWindow()

-void faztransferencia(Global a) [(10)]

-Jogador jogadoresNomeIgualNaEquipa(Global a, String nome, Equipa e) [(11)]

-Jogador jogadoresNomeIgual(Global a, String nome)

- Equipa equipasNomeIgual(Global a, String nome) [(12)]

- ArrayList<Jogador> pertenceequipatitular(ArrayList<Jogador> jog) [(13)]

-void simularJogoRegistado(Global a) [(14)]

-void simulajogoagora(Global a) [(15)]

-void simulajogo(Global a) [(16)]

-void volta(Global a) [(17)]

- ArrayList<Jogador> jogadorestitulares(Global a, Equipa equipa)

-void equipatitular(Global a, Equipa equipa) [(18)]

-Map<String,Jogador> substituicoes(Global a, Equipa equipa) [(19)]

-Jogo registarJogo(Global a) [20]

-void adicionarjogequipa(Global a) [21]

-void addjogequipa(Global a, Equipa equipa)

-ArrayList<Equipa> Champions (Global a)

-void menu(Global a)

[1]Método que imprime no ecrã o menu inicial da aplicação e nos permite escolher uma das opções desse Menu;

[2]Método que nos permite registar uma nova equipa, e associar novos jogadores a essa equipa;

[3]Método que permite ver todas as equipas já existentes, incluindo, ver uma equipa em especifico escolhida por quem está a usar a aplicação. Dentro disto, permite ainda realizar algumas ações dentro da equipa escolhida usando void acoesequipa;

[4]Método que permite alterar certos elementos da equipa, ou seja, deixa-nos remover e/ou adicionar jogadores à equipa;

[5]Método que nos permite identificar jogadores;

[6]Método que permite adicionar/criar mais jogadores. Usa registajogador;

[7]Todos estes são métodos que nos permitem criar novos jogadores e escolher os seus níveis de habilidade nos diferentes campos associados a cada posição;

[8]Método que permite registar um novo jogador, e que usa um dos métodos referidos em [7] dependendo da posição do jogador que queremos criar;

[9]Método que permite ver todos os jogadores existentes, e tal como [3], dá-nos a escolher de ver um jogador em particular;

[10]Método que permite realizar transferências de jogadores entre equipas;

[11]Método usado para identificar um jogador, quando, numa equipa, há dois jogadores com o mesmo nome, e procede para identificá-los pelo número que estes tem na camisola;

[12]Método que identifica uma equipa pelo seu número de id quando há mais do que uma com o mesmo nome;

[13]Método que verifica se um jogador pertence à equipa titular daquela em que se encontra;

[14]Método que apresenta os jogos que se encontram já registados e nos dá a oportunidade de observar a simulação do jogo que for escolhido;

[15]Método que nos permite escolher duas equipas e observar a simulação de um jogo entre elas;

[16]Método que simula jogos e utiliza os métodos [16], [17], [21],e também addJogo() que pode ser visto em Champions;

[17]Método que é usado quando nos é dada a opção de voltar ao menu inicial;

[18]Método que vai identificar os jogadores de uma equipa que vão ser jogadores titulares. A escolha pode ser feita por quem está a usar a aplicação ou aleatoriamente pelo programa;

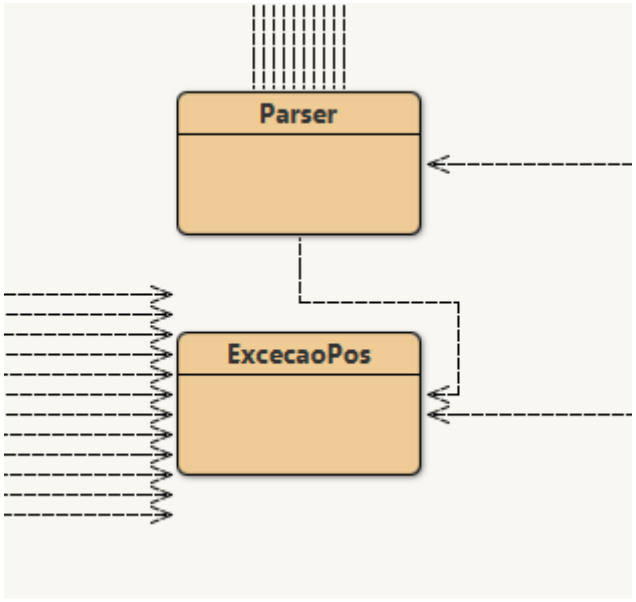
[19]Método que trata das substituições que acontecem numa equipa durante um jogo. Começa por perguntar quantas substituições se quer realizar (de zero a três), e procede para perguntar quais os jogadores que vão ser substituídos e os que vão entrar nos seus lugares

[20]Método que nos permite registar um novo jogo. Começa por pedir o nome das equipas que vão competir, e pode também a data em que o jogo vai ocorrer;

[21]Método que permite adicionar um jogador a uma equipa;

O funcionamento de todos os diferentes métodos mencionados neste tópico podem ser observados em **3.Descrição da Aplicação**.

2.4 Classes Auxiliares



Nas classes auxiliares vamos falar de 2 classes, sendo estas a ExcecaoPos e o Parse.

Sabendo que exceções se trata de um evento que ocorre durante a execução do programa que interrompe a sucessão normal de instruções, temos que a ExcecaoPos se trata das exceções em relação à posição dos jogadores no campo, ou seja, é usada quando a posição mencionada não corresponde a defesa, avançado, lateral, guarda-redes ou médio.

A outra classe auxiliar que temos presente é o Parse, cujos **métodos de classe** são:

-static void parse(Geral tudo)

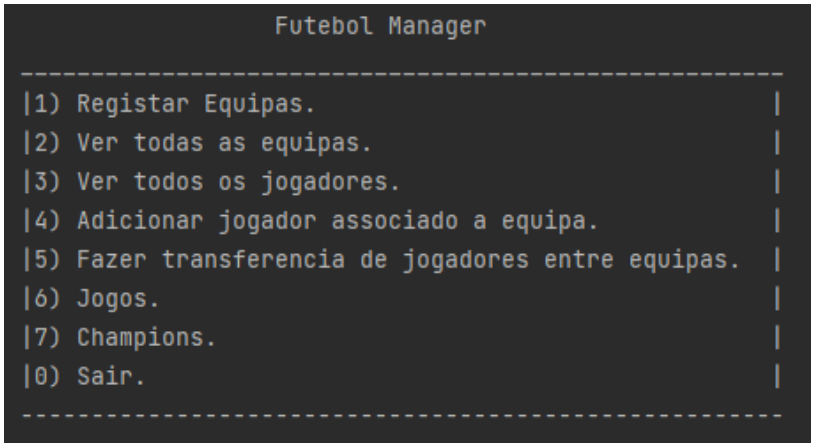
-static List<String> lerFicheiro(String nomeFich)

3. Descrição da aplicação

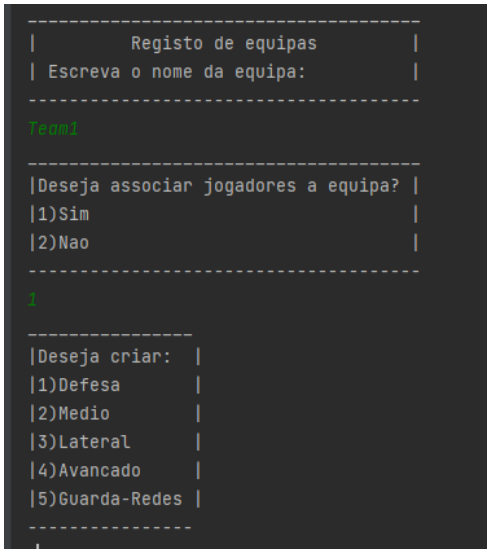
Neste tópico vamos abordar o funcionamento da aplicação, ou seja, vamos falar do menu inicial e de todas as opções com que somos deparados quando trabalhamos com esta; essencialmente, vamos demonstrar como a utilizar.

(Por referência foram criadas algumas equipas e jogadores para verificar o funcionamento do programa, e são essas que aparecem nas imagens)

A primeira coisa com que somos deparados ao abrir a aplicação é o Menu:



Ao optar por **1) Registar Equipas**, ficamos com a opção de criar um jogador, escolhendo o seu nome e a posição em que queremos que este jogue.



Após escolher uma das opções dadas, dependendo da posição escolhida, vão aparecer um conjunto de perguntas que varia de posição para posição, sobre o valor das habilidades do jogador, e para cada uma dessas questões, podemos dar um valor entre 0 e 100.

Ao escolher 1) Defesa, após dar um nome ao jogador que estamos a criar, as questões que aparecem tem a ver com a resistência, velocidade, destreza, impulsão, e também a habilidade para realizar cabeceamentos, remates e passes.

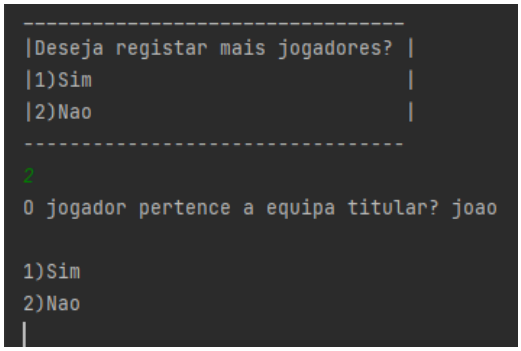
No caso em que se escolhe criar um jogador 2) Médio as questões são as mesmas da anterior, com a adicional que pergunta pelo valor de habilidade que o jogador tem para recuperar a bola.

Ao optar por 3) Lateral, voltamos a ter as mesmas questões que são apresentadas em 1), e somos também questionados sobre a capacidade que o jogador tem de realizar cruzamentos.

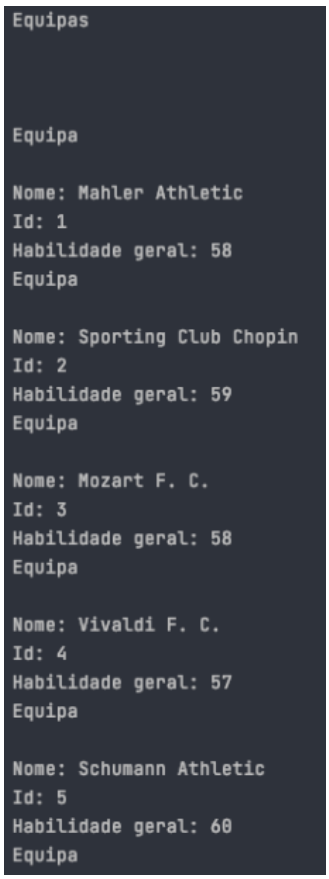
Em 4) Avançado as questões feitas são as mesmas que são feitas em 1).

Finalmente, em 5) Guarda-Redes temos também as questões apresentadas em 1) e também somos questionados pelo nível de elasticidade do jogador.

Quando já não queremos adicionar mais jogadores, somos questionados sobre o estatuto deste, ou seja, se é titular ou não.

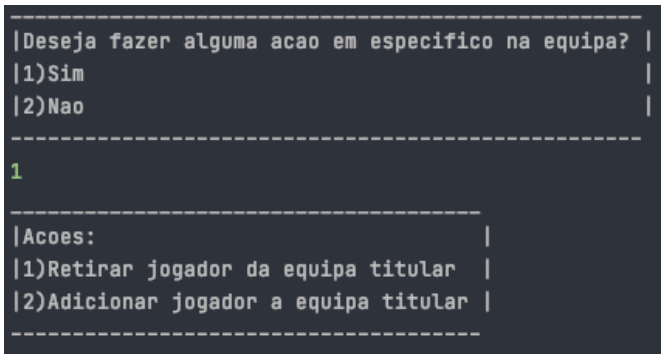


Ao optar por **2) Ver Todas as Equipas** no menu inicial, o programa vai-nos apresentar a lista de todas as equipas já existentes:



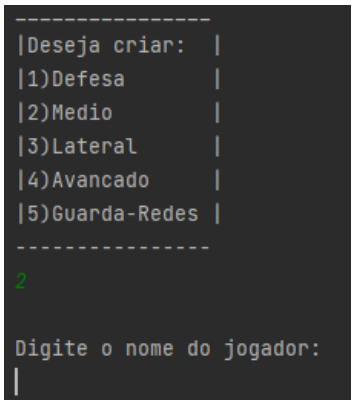
Após isto, pergunta-nos se há alguma equipa em especifico que queremos consultar, se a resposta for sim, esta vai pedir pelo nome da equipa e apresentar todos os dados que possui sobre a equipa escolhida, caso contrário dá-nos a opção de voltar ao menu inicial.

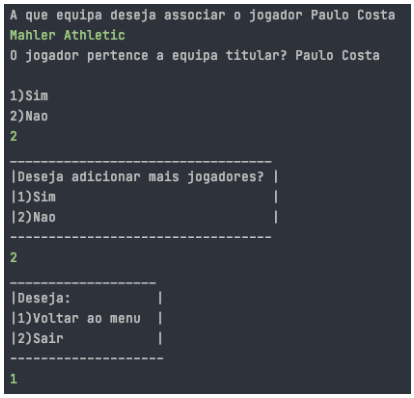
Depois de consultar a equipa temos ainda a opção de remover ou adicionar um jogador á equipa titular.



O mesmo vai acontecer se no menu inicial for selecionada a opção **3) Ver todos os jogadores**.

No menu inicial, ao escolher a opção **4) Adicionar jogador associado a equipa**, vamos ter a mesma opção que temos em **1)Registar Equipas** de criar ou um Defesa, um Médio, um Lateral, um Avançado ou um Guarda-Redes; após decidir os valores das diferentes habilidades do jogador, podemos escolher a equipa a que queremos que este pertença.





Se optarmos por **5) Fazer transferência de jogadores entre equipas**, o programa pede pelo nome da equipa de que o jogador vai sair e o da equipa para que tenciona que o jogador vá, e também pelo nome do jogador e efetua a transferência.

```
A que equipa pertence o jogador a ser transferido
Mahler Athletic

Para que equipa e que o jogador vai
Sporting Club Chopin

Nome do jogador:
Eduardo Costa de Magalhaes
O jogador pertence a equipa titular? Eduardo Costa de Magalhaes

1)Sim
2)Nao
2
--------------------
|Deseja:           |
|1)Voltar ao menu  |
|2)Sair            |
--------------------
```

Em **6) Jogos,** vamos ter 4 opções:

```
-----------------------------------
|      Simular jogo amigavel      |
|1)Registar um jogo               |
|2)Realizar um jogo ja registado  |
|3)Historico de jogos realizados  |
|4)Realizar um jogo amigavel agora |
-----------------------------------
```

Ao optar por 1)Registar um jogo, a primeira coisa a fazer é escolher uma equipa "casa",

```
Nome da equipa da equipa da casa:
Mahler Athletic

Identifique os jogadores que irao jogar na equipa Mahler Athletic
-----------------------------------
|Deseja:                          |
|1)Identificar os jogadores       |
|2)Fazer a escolha automaticamente |
-----------------------------------
2

----------------------------
|Qual tática pretende usar? |
|1)4-3-3                    |
|2)4-4-2                    |
|3)3-5-2                    |
----------------------------
1
```

O programa procede por mostrar os elementos que fazem parte da equipa selecionada, tanto os titulares como os suplentes, e de seguida pergunta quantas substituições deseja fazer durante o jogo, caso seja mais que zero, pede para selecionar quais os jogadores que pretende substituir. O processo repete para a equipa que é selecionada como visitante. E a partir daí vai acontecer o jogo.

Se escolhermos 2)Realizar um jogo já registado, no ecrã vão aparecer os jogos que foram registados no ponto anterior, e pede-nos para selecionar o nome das equipas que se vão enfrentar, a data do jogo, e dá-nos a opção de observar uma simulação.

```
-------------------------
|Deseja ver a simulacao? |
|1)Sim                   |
|2)Nao                   |
-------------------------
1
Mahler Athletic vs Sporting Club Chopin
```

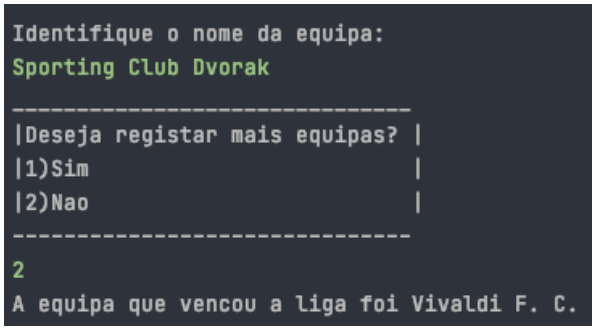
```
Jogo a decorrer
minuto: 0

Jogo a decorrer
minuto: 1

Jogo a decorrer
minuto: 2

Jogo a decorrer
minuto: 3

Jogo a decorrer
minuto: 4

Jogo a decorrer
minuto: 5
A equipa Sporting Club Chopin esta a atacar
Os defesas (Mahler Athletic) recuperam a bola.

A equipa Sporting Club Chopin vai realizar um lancamento lateral
Fantastico lancamento da equipa Sporting Club Chopin
Os avançados (Sporting Club Chopin) progredem e ganham espaço no campo
O Eduardo Costa de Magalhaesrealizou uma otima defesa!
```

Quando escolhemos a opção 3)Histórico de jogos já realizados, no ecrã vão aparecer todos os jogos realizados até aquele momento.

```
Historico de Jogos:

Dia: 2021-03-30
Jogo:
Sporting Club Shostakovich vs Mendelssohn F. C.
Resultado: 0-0


Dia: 2021-04-02
Jogo:
Mozart F. C. vs Sporting Club Dvorak
Resultado: 2-2


Dia: 2021-03-14
Jogo:
Debussy Athletic vs Stravinsky Athletic
Resultado: 5-2


Dia: 2021-02-05
Jogo:
Schumann Athletic vs Beethoven F. C.
Resultado: 2-1
```

Por fim, ao escolher a opção 4)Realizar um jogo amigável agora, vai acontecer aquilo que aconteceu nas outras opções, ou seja, vamos inserir o nome da equipa de casa, bem como os jogadores e a tática que pretendemos usar, e também o número de substituições; e fazemos o mesmo para a equipa visitante. Após isto somos dados a opção para ver uma simulação do jogo.

Por ultimo, no menu inicial, temos **7) Novo torneio Champions**. Ao selecionar esta opção, o programa pergunta-nos se queremos registar equipas, ao responder que sim, temos de colocar o nome das que queremos que façam parte do torneio; quando não queremos adicionar mais equipas, este diz-nos qual das selecionadas sai vitoriosa.



4. Conclusão

Bem, em jeito de conclusão achamos que este trabalho foi desafiante, no sentido em que para além de ter dado para abordar e perceber ainda melhor todos os conceitos abordados ao longo da disciplina, à medida que iramos avançando no projeto e a ter progressos, encontramos também vários obstáculos, os quais nos obrigaram a também uma pesquisa sobre certos aspetos e os quais conseguimos ultrapassar, por vezes com explicações entre os colegas que pertencem a este grupo. Acabou por ser uma forma bastante criativa e, até um certo nível, divertida de abordar os diversos temos que esta disciplina aborda. Dizemos até um certo nível uma vez que em certos momentos tornou-se frustrante não acertar em tudo à primeira.

Em relação ao projeto que foi obtido no final, achamos que conseguimos alcançar os requisitos pretendidos pelos professores, e na opinião do grupo, achamos que se obteve um jogo que é, de facto, muito adorado por um dos elementos do grupo, muito bem conseguido e o qual, no final, é daqueles projetos que se pode dizer que o esforço colocado foi recompensado e esperemos que os professores concordem também.

Damos assim, como concluído o relatório relativo ao projeto da UC- Programação Orientada aos Objetos 2020/2021, por parte do Grupo 63, composto por:

Ana Silva
Paulo Costa
Rui Baptista