Universidade do Minho

TripleSFootball

Trabalho prático da UC Programação Orientada a Objetos

94166

Samuel de Almeida Simões Lira

93253

David Alexandre Ferreira Duarte

81667

Daniel José Coutinho Gonçalves de Faria

Índice

Introdução

No âmbito da unidade curricular Programação Orientada a Objetos foi proposto aos alunos que criassem um Sistema de gestão e simulação de equipas de um determinado desporto. No caso desta solução o desporto escolhido foi o futebol.

A aplicação desenvolvida permite, de acordo com o solicitado nos campos 2.1 e 2.2 do enunciado, as seguintes funções:

- Criar Jogadores;

- Criar Equipas;

- Associar Jogadores a Equipas;

- Consultar Jogadores;

- Consultar Equipas;

- Transferência de um jogador para uma nova equipa e consequente registo em historial;

- Cálculo do resultado de um jogo entre duas equipas (em função de um cálculo baseado na habilidade do 11 de cada equipa);

- Definição das equipas titulares;

- Programação das substituições (efetuadas no intervalo);

- Gravar o estado de jogo em ficheiro;

- Carregar o estado de jogo de um ficheiro.

Descrição da solução

Arquitetura

A solução desenvolvida está dividida em dois “packages”:

- Gestão – De onde farão parte todas as classes relacionadas com a gestão dos dados do jogo. Criação de elementos tanto de jogador como de equipa;
- Main – Que contém as classes relacionadas com o estado, execução e carregamento do jogo.

No *package* Gestão podem encontrar-se as seguintes classes:

- Jogador
- Equipa
- Jogo
- Avancado
- Defesa
- GuardaRedes
- Lateral
- Medio

No *package* Main podem encontrar-se as seguintes classes:

- Crono
- Main
- Menu
- Parser
- TripleSFootball
- UserInterface

Package gestão

Jogador

Esta é uma classe abstrata e os seus atributos são a base da composição de qualquer jogador em TripleSFootball.

Cada jogador possui as seguintes variáveis privadas:

- nome :: String
- camisola :: int
- velocidade :: int
- resistência :: int
- destreza :. int
- impulsão :: int
- jogoDeCabeca :: int
- remate :: int
- capacidadeDePasse :: int
- historial :: List<String>

Em Jogador é ainda possível encontrar métodos de construção (vazio e parametrizado), *getters* e *setters* para todas as variáveis de instância, métodos *clone(), toString()* e *equals()*.

Para além destes existem ainda dois métodos abstratos: *calculaHabilidade()* e *jogadorParaLinha()*. O método *calculaHabilidade()* é abstrato porque cada tipo de jogador (Defesa, Médio, Lateral, e.t.c) tem atributos únicos dele e assim sendo cada posição terá a sua habilidade calculada de forma diferente.

O método *jogadorParaLinha()* é um método de auxílio à gravação do estado de jogo e a razão de ser abstrato é a mesma razão que está na origem do *calculaHabilidade()*, diferente número de atributos requerem formas diferentes de serem escritos em ficheiro os dados de um Jogador.

Avancado

A classe *Avancado* é uma *subclass* de *Jogador* e as suas variáveis de instância são relativas aos pesos que os atributos de um avançado têm no seu cálculo de habilidade. Estas variáveis estão presentes em todas as subclasses de *Jogador* embora possam existir em quantidade diferente consoante a posição em questão.

Relativamente aos métodos existem os todos os métodos base tal como em Jogador (*clone(), toString(), equals()* e construtores), existe o método *parse(input :: String)* que auxilia na adição de um *Avancado* (neste caso) ao estado do jogo, *jogadorParaLinha()* que traduz a informação do *Avancado* numa linha passível de ser lida de volta para um novo estado de jogo e por fim o método *calculaHabilidade()* que como já foi explicado anteriormente serve para calcular a habilidade do

jogador em causa tendo em conta o valor das suas características (*velocidade, destreza, remate, e.t.c)* e os pesos relativos a essas características.

Defesa

Em tudo semelhante à classe *Avancado.*

GuardaRedes

Em tudo semelhante à classe *Avancado* com a excessão de ter mais duas varíaveis de instância, a *elasticidade* e consequentemente o seu peso (*pElasticidade)*.

Lateral

Em tudo semelhante à classe *Avancado* com a excessão de ter mais duas varíaveis de instância, a *cruzamento* e consequentemente o seu peso (*pCruzamento)*.

Medio

Em tudo semelhante à classe *Avancado* com a excessão de ter mais duas varíaveis de instância, a *recuperacao* e consequentemente o seu peso (*pRecuperacao)*.

Package main

Parser

A classe *Parser* é a "ponte" entre um estado de jogo guardado em ficheiro e um estado de jogo na aplicação com todos os objetos criados.

Esta classe possui três variáveis de instância, são elas:

- equipas :: Map<String,Equipa>
- jogadores :: Map<String,Jogador>
- jogos :: List<Jogo>

A classe em questão não possui construtores nem métodos *clone(), toString() e quals()* mas tem *getters* para cada variável.

lerFicheiro(path :: String) é um método que permite armazenar numa lista de *String* a informação relativa a um estado de jogo

Contudo a aplicação possibilita também a gravação de um estado de jogo e aí entram estes dois métodos: *estadoParaLista(tsf :: TripleSFootball)* e *escreverFicheiro(path :: String, tsf :: TripleSFootball)*. *estadoParaLista* é um método que permite pegar num estado de jogo, com todos os seus jogadores, equipas e jogos, e com o auxílio de métodos de *Jogador* referidos anteriormente transformar este estado numa lista de *String* a ser usada pelo método *escreverFicheiro* que irá escrever linha a linha num ficheiro de texto o estado atual do jogo.

O método *parse(path :: String)* tem como objetivo criar a partir da lista que retorna o método *lerFicheiro* criar os objetos de *Jogador, Equipa e Jogo* transformando toda esta informação num estado de jogo na aplicação.

TripleSFootball

TripleSFootball é a classe que representa o estado do jogo numa determinada altura. Nesta classe é encontrada informação relativa a cada objeto do jogo em tempo real.

Eis as variáveis de instância desta classe:

- equipas :: Map<String,Equipa>
- jogadores :: Map<String,Jogador>
- jogos :: List<Jogo>

Esta classe possui os seguintes métodos de instância:

- addJogador(j :: Jogador)
- adicionarJogo(j :: Jogo)
- existeJogador(nome :: String)
- existeEquipa(nome :: String)

- existeJogadorNaEquipa(nomeJogador :: String, nomeEquipa :: String)
- adicionaEquipa(equipa :: Equipa)
- transfereJogador(nomeJogador :: String, nomeEquipaDestino :: Equipa)

Para além destes possui ainda os métodos *getters* e *setters*.

Menu

A classe *Menu* é a vista desta aplicação.
A partir dela é possível uma interação com o utilizador com base em listas de opções com números associados que permitem uma utilização intuitiva desta.

Eis as variáveis de instância desta classe:

- opcoes :: List<String>
- disponivel :: List<MenuPreCondition>
- handlers :: List<MenusHandler>

O construtor de um objeto de *Menu* recebe uma lista de *String* que são as opções a imprimir no terminal do utilizador e ficam alocadas em *opcoes.*

Como métodos de instância existem:

- run() :: void
- setPreCondition(i :: int, b : MenuPreCondition) :: void
- setHandler(i : int, h :: MenuHandler) :: void
- show() :: void
- readOption() :: int

Existem ainda duas interfaces: *MenuHandler* e *MenuPreCondition* sendo a primeira para ligar uma opção do utilizador a um método na aplicação e a segunda para validar ou não a impressão de uma opção no terminal.

UserInterface

UserInterface é o controlador da aplicação.
Faz gestão das interações do utilizador com o estado do jogo.

Esta classe tem as seguintes variáveis de instância:

- p :: Parser
- tsf :: TripleSFootball
- scin :: Scanner

Como construtor tem apenas o construtor vazio.

Eis os métodos de instância desta classe:

- run() :: void
- comecarJogo() :: void
- criarJogador() :: void
- criaGuardaRedes() :: void
- criaMedio() :: void
- crialateral() :: void
- criaJogadorElse() :: void
- criarEquipa() :: void
- iniciarJogo() :: void
- calculaProbMarcar(eCasa :: List<Jogador>, eFora :: List<Jogador>) :: int
- listarEquipas() :: void
- listarJogadores() :: void
- listarJogadoresEquipa() :: void
- transferirJogador() :: void
- salvarJogo() :: void

Alguns destes métodos são auxiliares e poderiam estar implementados na classe *TripleSFootball*.

LinhaIncorretaException

Exceção criada com vista no controlo de erros de linhas lidas de ficheiro.

Main

Esta classe tem como única função criar um *UserInterface* e chamar o método *run()* desse *UserInterface* iniciando assim a aplicação.

Diagrama de Classes

Descrição da Aplicação

No menu inicial são apresentadas duas opções: O utilizador pode decidir entre começar um novo jogo (a partir do estado facultado pelos docentes) ou carregar um jogo (a partir de um estado gravado previamente).

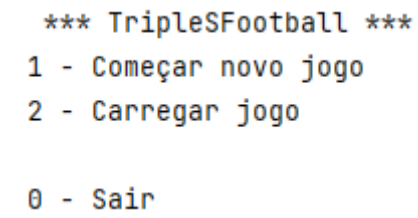

Figura 2 – Menu inicial

Após escolher o estado com o qual pretende jogar, é apresentado ao utilizador a seguinte lista de funcionalidades:

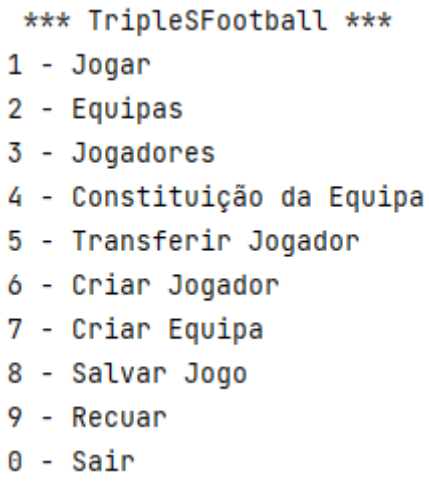

Figura 3 - Menu Principal

Se optar pela opção 1, Jogar, o utilizador é levado para um estado de pré-jogo onde deverá escolher as equipas que se irão defrontar assim como os onzes iniciais e ainda substituições se assim o entender.

```
Valor: 50,66 - Schumann Athletic
Valor: 49,14 - Stravinsky Athletic
Valor: 48,10 - Bach F. C.
Valor: 51,28 - Debussy Athletic
Valor: 46,48 - Mozart F. C.
Valor: 45,89 - Handel Athletic
Valor: 49,36 - Mendelssohn F. C.
Valor: 51,15 - Sporting Club Shostakovich
Valor: 49,99 - Sporting Club Schubert
Valor: 48,53 - Sporting Club Chopin
Valor: 46,87 - Mahler Athletic
Valor: 46,95 - Bartok F. C.
Valor: 47,11 - Beethoven F. C.
Valor: 49,63 - Sporting Club Prokofiev
Valor: 45,60 - Vivaldi F. C.
Valor: 47,42 - Sporting Club Dvorak
Valor: 47,81 - Brahms F. C.
Valor: 50,72 - Wagner Athletic
```

Figura 4 - Lista de Equipas Para o Confronto

```
44 - Alexandre Eduardo Vieira Martins Habilidade: 73.97
29 - Joao Pedro Fontes Delgado Habilidade: 51.550000000000004
12 - Pedro Alexandre Coutinho Goncalves de Sousa Habilidade: 34.74
28 - Alexandre Silva Martins Habilidade: 43.09
34 - Miguel Angelo Ferreira Fernandes Habilidade: 37.300000000000004
7 - Guilherme de Araujo Soares Habilidade: 50.410000000000004
38 - Luis Miguel Teixeira Fernandes Habilidade: 54.56
42 - Jose Joaquim Machado Barbosa Habilidade: 64.67
33 - Rui Pedro Chaves Silva Lousada Alves Habilidade: 53.440000000000005
4 - Daniel Torres Neves Habilidade: 48.31
2 - Cristiano Jose Goncalves Neiva Pereira Habilidade: 43.03
46 - David Alexandre Ferreira Duarte Habilidade: 64.04
9 - Andre Oliveira de Paula Boechat Morandi Habilidade: 56.31999999999999
25 - Beatriz Ribeiro Terra Almeida Habilidade: 55.349999999999994
15 - Sara Lima Pereira Habilidade: 46.06
37 - Pedro Santos Arezes Costa Habilidade: 48.72
27 - Pedro Gabriel Fernandes Pereira Habilidade: 42.97
14 - Ricardo Manuel Soares Peixoto Habilidade: 51.660000000000004
45 - Joao Manuel Silva de Amorim Habilidade: 44.410000000000004
47 - Tiago Andre Oliveira Leite Habilidade: 49.830000000000005
```

Figura 5 - Lista De Jogadores de Uma das Equipas Selecionadas

Tendo agora acesso aos valores de habilidade dos jogadores e aos números das suas camisolas o utilizador pode agora escolher os jogadores para formarem o 11 inicial e ainda programar as substituições.

Após este processo é apresentado ao utilizador o resultado da partida.

```
Wagner Athletic 3 : 0 Brahms F. C.
```

Figura 6 - Apresentação do Resultado de uma Partida

Outra funcionalidade do menu principal é a listagem de todas as equipas do jogo onde para além dos seus nomes é também apresentado o valor de habilidade de cada uma baseado no valor dos jogadores que as integram.

```
Valor: 50,66 - Schumann Athletic
Valor: 49,14 - Stravinsky Athletic
Valor: 48,10 - Bach F. C.
Valor: 51,28 - Debussy Athletic
Valor: 46,48 - Mozart F. C.
Valor: 45,89 - Handel Athletic
Valor: 49,36 - Mendelssohn F. C.
Valor: 51,15 - Sporting Club Shostakovich
Valor: 49,99 - Sporting Club Schubert
Valor: 48,53 - Sporting Club Chopin
Valor: 46,87 - Mahler Athletic
Valor: 46,95 - Bartok F. C.
Valor: 47,11 - Beethoven F. C.
Valor: 49,63 - Sporting Club Prokofiev
Valor: 45,60 - Vivaldi F. C.
Valor: 47,42 - Sporting Club Dvorak
Valor: 47,81 - Brahms F. C.
Valor: 50,72 - Wagner Athletic
```

Figura 7 - Opção 2 do Menu Principal (Lista de Equipas)

Da mesma forma que é possível mostrar as equipas do jogo é também possível mostrar os jogadores e correspondentes valores de habilidade individuais.

```
Valor: 40,85 - Diogo Manuel Brito Pires
Valor: 52,02 - Joao Domingos Pereira Barbosa
Valor: 69,91 - Silvia Maria Acosta Roca
Valor: 38,23 - Pedro Miguel Marques Fernandes
Valor: 51,22 - Francisco Jose Martinho Toldy
Valor: 44,18 - Joao Antonio Ribeiro Alves
Valor: 42,33 - Cristiana Tamborino Ribeiro
Valor: 44,65 - Goncalo Couto dos Santos
Valor: 42,50 - Bruno Filipe Miranda Pereira
Valor: 42,83 - Joao Ricardo Rodrigues Nogueira
Valor: 54,86 - Diogo da Costa e Silva Lima Rebelo
```

Figura 8 - Opção 3 do Menu Principal (Listagem de Jogadores) – Excerto

Se o utilizador estiver interessado em saber a constituição de uma equipa isso também é possível fazer com a quarta opção do menu principal – Constituição da Equipa.

```
Equipa:
Wagner Athletic
44 - Alexandre Eduardo Vieira Martins - Valor: 73,97
29 - Joao Pedro Fontes Delgado - Valor: 51,55
12 - Pedro Alexandre Coutinho Goncalves de Sousa - Valor: 34,74
28 - Alexandre Silva Martins - Valor: 43,09
34 - Miguel Angelo Ferreira Fernandes - Valor: 37,30
7 - Guilherme de Araujo Soares - Valor: 50,41
38 - Luis Miguel Teixeira Fernandes - Valor: 54,56
42 - Jose Joaquim Machado Barbosa - Valor: 64,67
33 - Rui Pedro Chaves Silva Lousada Alves - Valor: 53,44
4 - Daniel Torres Neves - Valor: 48,31
2 - Cristiano Jose Goncalves Neiva Pereira - Valor: 43,03
46 - David Alexandre Ferreira Duarte - Valor: 64,04
9 - Andre Oliveira de Paula Boechat Morandi - Valor: 56,32
25 - Beatriz Ribeiro Terra Almeida - Valor: 55,35
15 - Sara Lima Pereira - Valor: 46,06
37 - Pedro Santos Arezes Costa - Valor: 48,72
27 - Pedro Gabriel Fernandes Pereira - Valor: 42,97
14 - Ricardo Manuel Soares Peixoto - Valor: 51,66
45 - Joao Manuel Silva de Amorim - Valor: 44,41
47 - Tiago Andre Oliveira Leite - Valor: 49,83
```

Figura 9- Opção 4 do Menu Principal (Constituição de uma Equipa)

Caso queira transferir um jogador de uma equipa para a outra pode selecionar a opção 5 do menu. Esta opção pode ainda ser utilizada para associar um jogador sem equipa a uma equipa.

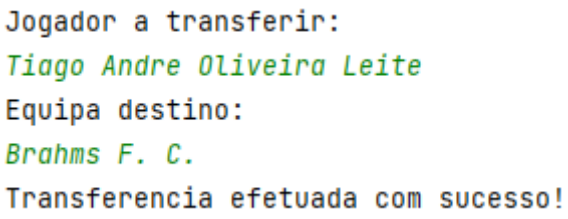

Figura 10 - Opção 5 do Menu Principal (Transferência de um Jogador)

Para além destas funcionalidades é ainda possível criar um jogador de raiz onde o utilizador pode escolher os valores dos atributos do jogador em questão assim como o seu nome, número e posição.

O mesmo processo se aplica à criação de uma equipa porém estas serão sempre criadas vazias e a única coisa que o utilizador pode escolher é o seu nome. Mais tarde poderá transferir jogadores para esta nova equipa, quer pertencentes a outras equipas quer criados pelo utilizador.

Finalmente o utilizador tem a opção de gravar o estado de jogo atual onde após selecionar essa funcionalidade lhe é pedido que escolha o nome a dar à gravação sem espaços para que mais tarde se decidir carregar este estado lhe seja apresentado esse nome de gravação.

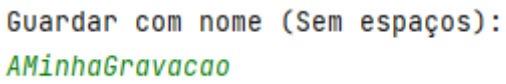

Figura 11 - Opção 8 do Menu Principal (Gravação do Estado de Jogo)

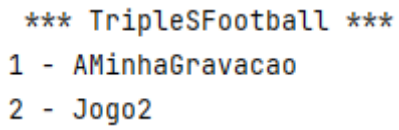

Figura 12 - Apresentação da Gravação do utilizador para Carregamento

Conclusões

Com a realização deste projeto o grupo assentou conhecimento sobre programação orientada a objetos adquiridas nas aulas da Unidade Curricular como por exemplo as técnicas de encapsulamento.

O grupo consegui implementar as funcionalidades pedidas à exceção da simulação de um jogo.

Com vista no que foi implementado o grupo tem consciência de que há aspetos que poderiam estar melhores tais como uma melhor separação de vista e controlador ou até mesmo um melhor reaproveitamento de código.