UNIVERSIDADE D
COIMBRA

Faculdade de Ciências e Tecnologias

Programação Orientada aos Objetos - Licenciatura em Engenharia Informática

POO Trivia

Trabalho Realizado por:
Nuno Batista e Diogo Joaquim

Dezembro, 2023

1 Objetivo e Introdução

O principal objetivo deste relatório é a apresentação das soluções obtidas durante o processo de desenvolvimento de um jogo de *trivia* utilizando métodos de programação orientada a objetos.

O jogo é constituído por 5 perguntas selecionadas aleatoriamente de várias categorias, sendo estas: "ciência", "arte" e "desporto". A categoria "desporto", tem 3 subcategorias: "ski", "natação" e "futebol". Cada categoria e subcategoria tem características diferentes (tanto o formato das mesmas, método de cálculo das suas pontuações, etc.).

No final de cada jogo, deverá também ser exibida uma lista com as 3 melhores pontuações previamente obtidas, juntamente com os respetivos nomes dos jogadores e as datas e horas.

Quanto à interação com o jogador, esta realizar-se-á através de uma interface gráfica feita com o *java swing* inspirada na estética dos terminais dos computadores da década de 70 (Imagem 1.1).

Imagem 1.1: IBM 3476

1.1 Estrutura geral

Tendo em conta os objetivos do projeto, a estrutura decidida foi a ilustrada no final do relatório (Imagem 4.2).

Adicionalmente, o método *main* está inserido na classe *Window*, usada para o GUI, ao contrário da convencional inserção numa classe à parte. Esta decisão foi tomada devido ao facto de a única função do método main ser a criação de um objeto do tipo *Window*, tal não justifica a criação de uma nova classe.

Dentro da pasta *code*, onde está o código fonte das classes criadas, há também uma subpasta, *gamefiles* que não só contém o ficheiro com as perguntas (*pootrivia.txt*) e o ficheiro que acompanha os dados dos jogos previamente jogados (*gamesplayed.txt*), como também o logotipo e a fonte do jogo (*logo.png* e *ibmbiosfont.ttf*). Será na pasta *gamefiles* que serão guardados posteriormente os ficheiros que contém informação sobre os jogos jogados.

1.2 Visão geral das classes usadas

- *Option* para definir a descrição de cada opção do jogo bem como se ela está correta ou errada.
- *TriviaGame* para guardar informação sobre o jogo atual ou ler informação sobre jogos posteriores
- *Question*, uma classe abstrata que fará a gestão das perguntas. Desta herdam as seguintes classes:
  - *ArtsQuestion* para gerir métodos e atributos específicos a perguntas da categoria "artes".

- *ScienceQuestion* para gerir métodos e atributos específicos a perguntas da categoria “artes”.
- FootballQuestion para gerir métodos e atributos específicos a perguntas da subcategoria “futebol”.
- SkiQuestion para gerir métodos e atributos específicos a perguntas da subcategoria “ski”.
- SwimmingQuestion para gerir métodos e atributos específicos a perguntas da subcategoria “natação”.

● *FootballPlayer* para gerir os atributos de cada jogador de futebol, sendo esses o respetivo número da camisola e nome.

● *Window* para gerir o jogo em si e os elementos da *graphical user interface*, o método main está incluído nesta classe, como mencionado previamente.

● *FileManager* que contém os métodos para gerir os ficheiros do jogo.

2 Funcionamento

2.1 Menu principal

O programa começa por criar uma instância da classe *Window* e chamar o método *mainMenu*. Já dentro do método *mainMenu*, será carregado o logotipo do jogo da pasta (*gamefiles*) e criados dois botões, *quit* e *newGame*. Caso o utilizador pressione *quit*, o programa terminará, caso opte por pressionar *newGame*, será dado início ao jogo e será também chamada o método *setupGame* do *FileManager*, que vai ler as perguntas do ficheiro *pootrivia.txt*.

2.2 *Parsing* do ficheiros das perguntas

Cada pergunta no ficheiro é composta por um conjunto de 3 linhas, sendo a primeira o enunciado da pergunta, a segunda define a categoria a que a mesma pertence, e a terceira contém a pontuação da pergunta antes da aplicação dos majorantes, as opções, (sendo a primeira opção sempre a correta) e um facto sobre a resposta certa, todos estes separados por “;”. É de notar que caso se trate de uma pergunta de ciência, as primeiras 5 opções constituem o conjunto de opções da pergunta normal e as últimas 5 o conjunto de opções da pergunta facilitada .

As categorias, definidas na segunda linha, seguem a seguinte notação: “$” corresponde a ciência, “&” corresponde a artes e “@” corresponde a desporto, sendo “@1” futebol, “@2” ski e “@3” natação. Dependendo da categoria da pergunta, será chamada um método diferente para fazer o *parsing* do formato de pergunta correto e guardá-lo num array de perguntas.

Após a leitura, o array será baralhado e serão escolhidas as 5 primeiras perguntas do array para serem perguntadas ao longo do jogo.

2.3 Exibição das perguntas

A pergunta a ser apresentada é definida pela fase do jogo onde o jogador se encontra, essa fase é indicada pelo atributo *gameStage*, caso o valor de *gameStage* seja menor que 2, será apresentada a respetiva versão fácil da pergunta.

Se o utilizador selecionar a opção correta, o índice *gameStage* do array *correctIndices* associado ao *TriviaGame* atual, será mudado para *true*. Após a seleção da opção, será chamado o método *afterQuestion*. Que não vai apenas revelar se o jogador acertou ou não, como vai também apresentar um facto sobre a pergunta, incrementar o valor de *gameStage* e passar à próxima pergunta.

2.4 Final do jogo

Após terem sido respondidas 5 perguntas (ou seja, quando *gameStage* for igual a 5), será chamado o método *getPlayerName* para definir o nome do jogador para, posteriormente, apresentar no “Top 3”. De seguida será chamado o método *endGame*, esta guardará os dados do jogo atual num ficheiro .dat e o nome do ficheiro acabado de guardar, será concatenado no final do *gamesplayed.txt*. De seguida os ficheiros dos jogos previamente jogados, serão lidos, os scores

desses jogos serão calculados, e as 3 melhores pontuações serão extraídas e apresentadas num Top 3, juntamente com a pontuação obtida no jogo atual.

Finalmente, será dada ao jogador a opção de sair do jogo ou jogar novamente, que reiniciará o ciclo do jogo desde o menu.

2.5 Explicação de algumas soluções

Como este relatório tem um limite de páginas para a explicação do funcionamento do jogo, a descrição do ciclo de jogabilidade teve de ser extremamente resumida, no entanto o detalhamento das seguintes soluções foi considerado imprescindível.

2.5.1 Porque não usar uma classe pai *SportsQuestion?*

Atendendo ao facto das perguntas das 3 subcategorias de desporto não terem características em comum para além das já definidas na classe *Question*, não foi necessário criar uma classe pai *SportsQuestion*, o código tornou-se mais simples ao criar diretamente uma classe individual para cada subcategoria de "desporto". Apesar das subcategorias de ski e natação partilharem a característica de terem apenas verdadeiro ou falso como opções, esta não é relevante para a maneira como as opções são guardadas, visto que verdadeiro e falso não passam de objetos *Option* como em qualquer outra pergunta, adicionalmente, nenhuma das duas tem um modo fácil, portanto nem foi necessário definir *setEasyMode* em nenhuma das duas.

2.5.2 Cálculo das 3 melhores pontuações

Para fazer o cálculo das 3 melhores pontuações, o algoritmo usado consiste em iniciar 3 variáveis *TriviaGame*: *first*, *second* e *third*, todas elas, quando calculada a pontuação, vão devolver -1. De seguida, será iterado o array contendo todos os jogos. Sempre que o jogo a ser comparado de momento, tiver uma pontuação maior que o *first*, *second* será movido para *third*, *first* será movido para *second* e o jogo atual será movido para *first*. Se a pontuação do jogo atual for maior que o *second*, mas menor que o *first*, *second* será movido para third e o jogo atual será movido para *second*. Finalmente, se a pontuação do jogo atual for apenas maior que o *third*, o jogo atual será movido para *third*.

Foi escolhido este algoritmo em oposição ao método *sort* da classe *Arrays* para, idealmente, minimizar o cálculo de scores, como explicado a seguir.

Efetivamente, para calcular a pontuação de um jogo, é necessário iterar 5 perguntas por cada *TriviaGame*, a complexidade de tempo desta operação, é:

$$O(5)$$

Isto quer dizer que o algoritmo implementado, sendo n o número de jogos a analisar no pior cenário possível, ou seja, caso a pergunta a ser analisada em cada iteração tenha uma pontuação pior que a *first* e a *second* mas melhor que a *third* (porque serão necessárias 3 comparações) vai ter complexidade de tempo:

$$O(3(5n) = O(15n)$$

E no melhor caso possível, ou seja, caso o jogo atual seja sempre melhor que a *first,* será necessária apenas uma comparação e a complexidade de tempo será apenas:

$$O(5n)$$

Comparando agora ao método *sort* da classe *Arrays* cujo melhor caso cenário é, já tendo em conta as 5 iterações por pergunta faladas anteriormente:

$$O((5n)log(5n))$$

E o pior caso cenário, que acontece quando o pivot está nas extremidades do array, ou seja, quando o melhor ou pior elemento está no primeiro ou último índice do array será:

$$O((5n)^2)$$

Resolvendo as equações, mesmo caso aconteça o pior caso possível do algoritmo implementado, o que já é algo improvável de acontecer, será melhor que o melhor caso do método *sort* quando o jogo já foi jogado 4 vezes e melhor que o pior caso do método sort quando jogado apenas 1 vez:

$$15n < (5n)log(5n) \iff n > \frac{e^3}{5} \approx 4.0171$$

$$15n > (5n)^2 \iff n > 0.6$$

Para auxiliar ainda mais o algoritmo implementado, pode-se pensar que em teoria, quanto mais vezes o jogo é jogado, maior é a chance do jogador atual ter experiência com o jogo, colocando a sua pontuação em primeiro lugar, o que causaria o pior caso cenário do método *sort*.

3 Manual do Utilizador

Em cada jogo de POO Trivia, serão apresentadas 5 questões de várias categorias, para acumular pontos, o jogador deve escolher a opção correta para cada uma delas.

Para começar a jogar POO Trivia, basta executar a aplicação do jogo. No menu principal serão apresentadas duas opções: **Quit** e **New Game**, para sair do jogo, selecione **Quit**, para iniciar as 5 perguntas, selecione **New Game**.

Serão agora apresentadas 5 perguntas, sequencialmente, cada uma tem uma pontuação base definida, apresentada na parte superior do ecrã, que será posteriormente majorada de acordo com a categoria e subcategoria (Imagem 3.1).

Área	Características	Majoração da pontuação
Artes	-	x 10
Ciências	-	+ 5
Desporto	-	+ 3
Futebol	Subárea de Desporto.	+ 1
Natação	Subárea de Desporto.	+ 10
Ski	Subárea de Desporto.	x 2

Imagem 3.1: Majoração das perguntas

Cada categoria tem também, uma estrutura diferente de opções:

- Artes: 6 opções.
- Ciências: 5 opções.
- Futebol: 5 números de camisolas de jogadores.
- Ski e Natação: 2 opções (verdadeiro ou falso).

É também importante mencionar que as duas primeiras questões apresentadas serão modificadas para serem mais fáceis, cada categoria de pergunta sofre uma alteração diferente durante este processo:

- Artes: 3 das opções são removidas.
- Ciências: As opções são substituídas por um conjunto de opções mais fáceis.
- Futebol: Os números de camisolas são substituídas pelos nomes dos respetivos jogadores.
- Ski e Swimming: Não sofrem alterações

Após as 5 perguntas terem sido respondidas, é pedido ao jogador o seu nome (caso não escolha nome, este será substituído por "*No Name*"). A pontuação total do jogador, agora incluindo os majorantes mencionados anteriormente, será comparada à dos jogadores que jogaram previamente e colocada numa tabela com os 3 melhores jogos.

Finalmente, como no menu principal, serão dadas as opções **Quit** e **New Game**, para sair do jogo deve selecionar **Quit** e para voltar ao menu principal e jogar novamente selecione **New Game.**

4 Conclusão

Deste projeto, resultou, não só um entendimento mais profundo sobre programação orientada a objetos, como também um jogo de *trivia* bastante completo, que cumpre todos os objetivos colocados no início do projeto, com uma interface gráfica intuitiva e reminiscente dos terminais antigos (Imagem 4.1).

A estrutura final do projeto (Imagem 4.2), sofreu bastantes alterações desde a estrutura inicial, no entanto, todas elas foram justificadas e todas as soluções encontradas, apesar de não haver espaço para as descrever neste relatório, têm comentários no código fonte que explicam em detalhe o funcionamento das mesmas.

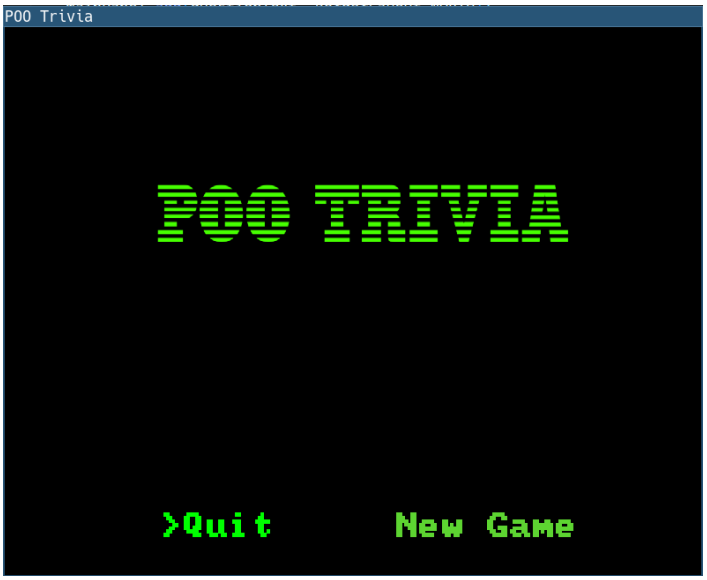

Imagem 4.1: Menu principal do POO Trivia

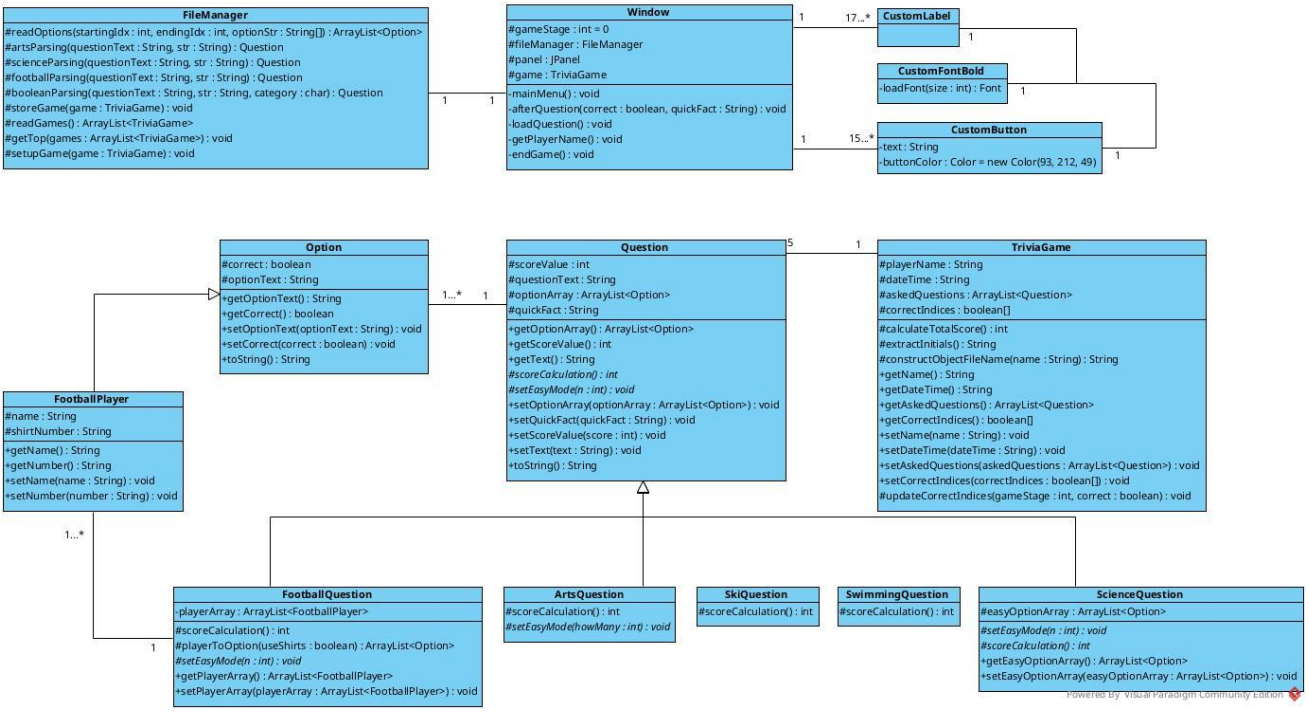

Imagem 4.2: Versão final do UML

5 Bibliografia

Informação sobre a complexidade de tempo do algoritmo Quick Sort: Quick Sort Analysis

Imagem 1.1: Old IBM terminal