isec Engenharia

Trabalho Prático

Primeira fase

4 EM LINHA

Programação Avançada

2020/21

Pedro Correia - 2018020558

Índice

Introdução

Este projeto foi desenvolvido no âmbito da Unidade Curricular de Programação Avançada, da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

O objetivo era desenvolver uma aplicação que simulasse o famoso jogo do 4 em Linha, com alguns minijogos intermédios que fornecem uma peça especial, capaz de eliminar todas as peças de uma coluna, aos vencedores desses minijogos. O jogo tem 3 modos distintos: Jogador vs Jogador, Jogador vs Computador e Computador vs Computador.

A estratégia adotada passou pela implementação do padrão de uma máquina de estados que gere o decorrer dos jogos.

É dada ao utilizador a possibilidade de gravar, recuperar e ver o replay de jogos, funções estas que não estão inseridas no contexto da máquina de estados, utilizando ficheiro binários gravados através de Serialização.

O utilizador também tem a possibilidade de desfazer as jogadas. Para esta funcionalidade implementou-se o padrão Memento, capaz de gravar *snapshots* de todas as jogadas efetuadas.

No decorrer dos jogos é gerado um *log* com toda a informação de cada jogada. Este *log* é mostrado ao utilizador quando vê um replay ou carrega um jogo.

Para diferenciar os tipos de jogador, optou-se pela criação de uma classe que implementa diferentes métodos para os dois tipos de jogador. O mesmo acontece para os minijogos, havendo também dois possíveis: minijogo de respostas matemáticas e minijogo de respostas de texto.

Diagrama da máquina de estados

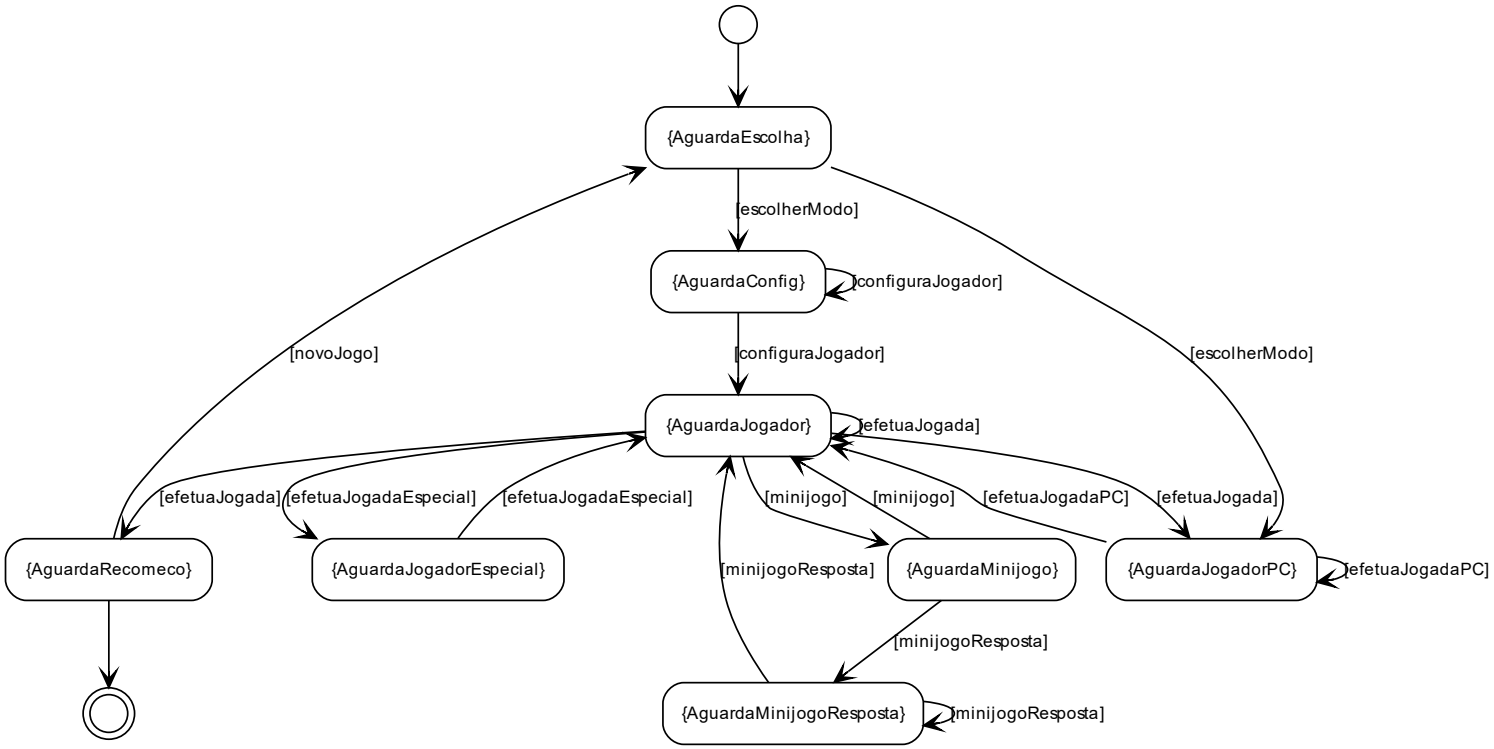

Figura 1 – Diagrama da máquina de estados

Diagrama de outros padrões

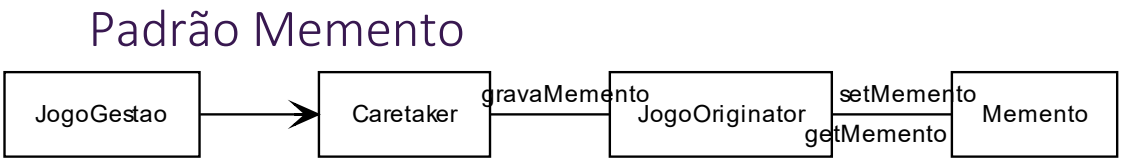

Figura 2 – Diagrama do Padrão Memento

Como podemos verificar pelo diagrama acima, a classe *Caretaker* não tem acesso aos métodos da classe *Memento*, por essa razão, apenas a classe *JogoOriginator* consegue alterar o estado interno dos *mementos*.

Descrição das classes

Principal

Esta classe inclui somente o método *main*, utilizado para iniciar a aplicação. Necessita de incluir uma referência para a classe *JogoGestao* e *UIText*, de modo a poder iniciar o jogo e a interface de texto.

UIText

A classe *UIText* representa a interface de comunicação com o utilizador, neste caso no formato consola. Esta tem uma referência para um objeto da classe *JogoGestao*, passado por argumento pela classe *Principal*.

O principal objetivo desta classe é comunicar com o utilizador, mostrando as ações que pode realizar e pedir que este as efetue. Para isso, apenas torna público o seu método *run*, chamado no método *main* da classe *Principal*.

Todos os restantes métodos não são acedidos fora da classe, pois esta apenas interage com a classe *JogoGestao*, nunca comunicando diretamente com a máquina de estados. Para comunicar com o utilizador de forma eficiente as ações que pode realizar, são utilizadas Enumerações (*enum Situacao)*, correspondentes a cada estado do jogo.

Jogo

A classe *Jogo* representa a comunicação entre o utilizador e a máquina de estados. É nesta classe que se encontra a referência para o estado do jogo e para os dados deste. Nesta classe é utilizado o método privado *setEstado* para alterar o estado do jogo. Esta classe também obtém dados importantes sobre o jogo, que são passados para a interface, como o *log*.

JogoGestao

Esta classe surge com a necessidade de implementar o padrão Memento. Nesta classe são instanciados os objetos da classe *Caretaker* e *JogoOriginator* responsáveis pela gestão dos *snapshots* das diversas fases do jogo. Esta classe delega os seus métodos para a classe *JogoOriginator*.

JogoOriginator

A classe *JogoOriginator* é responsável por implementar os métodos *getMemento* e *setMemento* da interface *IMemento*. "Esconde" ainda a classe *Memento*, onde é efetivamente feito a gravação e obtenção de *snapshots*. Esta funcionalidade é útil pois permite que a classe *Caretaker* não tenha acesso aos métodos da classe *Memento*.

O objetivo desta classe é a possibilidade de implementação da função *undo* para cada jogada efetuada. Esta classe comunica diretamente com a classe *Jogo*.

Situacao

Esta é uma Enumeração, responsável por devolver o estado atual da máquina de estados, de modo que a interface consiga disponibilizar a informação correta ao utilizador.

Util

Esta classe implementa métodos de gravação e recuperação de ficheiros binários. O seu objetivo é implementar as funcionalidades de gravação e carregamento de jogos, bem como a gravação e carregamento de replays.

AguardaEscolha

Esta classe representa o estado do jogo em que o utilizador necessita de escolher qual dos três modos pretende jogar.

AguardaConfig

A classe *AguardaConfig* representa o estado de configuração dos jogadores. Caso seja escolhido o modo de jogo Computador vs Computador, a máquina de estados nunca irá passar por este estado.

AguardaJogador

Nesta classe é implementado o método *efetuaJogada*, responsável por colocar uma ficha na coluna pretendida. Este é o estado do jogo em que se espera que o utilizador introduza a coluna onde pretende jogar, ou outras ações que pode realizar, como o *undo* ou gravação do jogo.

AguardaJogadorPC

Esta classe representa o estado de uma jogada de um jogador artificial. Foi considerado este estado pois é pedido ao utilizador para premir uma tecla de modo a avançar com o jogo. Nesta classe é implementado o método *efetuaJogadaPC*, onde é gerado aleatoriamente uma coluna para jogar, sem recorrer a nenhum algoritmo de inteligência artificial.

AguardaMinijogo

Nesta classe é esperado que o utilizador introduza se pretende realizar ou não o minijogo proposto.

AguardaMinijogoResposta

Nesta classe é pedido ao utilizador a resposta às perguntas do minijogo. Caso o utilizador ganhe, é oferecida uma peça especial

AguardaJogadorEspecial

Aqui é esperado que o utilizador introduza a coluna para a qual pretende eliminar as peças, ou seja, em que coluna deseja utilizar a sua peça especial.

AguardaRecomeco

Nesta classe é terminado o jogo atual, e dada a possibilidade ao utilizador de começar um novo jogo.

EstadoAdapter

É uma classe abstrata, que implementa métodos neutros da Interface *IEstado*. Serve de ponte para a comunicação entre a Interface e a implementação dos métodos em cada estado.

IEstado

É uma Interface, apenas declara métodos que serão implementados noutras classes. Declara todos os métodos usados pela máquina de estados.

Caretaker

Esta classe é responsável apenas por gravar e devolver os *mementos*, não tem informação acerca do objeto que grava. É implementado o método *gravaMemento* que é chamado pelo *JogoOriginator* quando é necessário gravar um *memento*.

IMemento

Esta interface apenas existe para a classe *Caretaker* poder guardar *mementos*, encontra-se vazia de modo que o *Caretaker* não tenha acesso aos seus métodos.

IMementoOriginator

Declara os métodos *getMemento* e *setMemento* que são implementados na classe *JogoOriginator*.

Dados

Nesta classe estão implementados os dados referentes ao jogo. Esta classe armazena as referências para os jogadores, o tabuleiro e os minijogos, assim como é onde é gerado o *log* de cada jogada.

Aqui são implementados os métodos que permitem a construção do jogo, como a escolha do modo e a configuração dos jogadores.

Esta classe comunica diretamente com a classe *Jogo*, que invoca alguns dos seus métodos, e com a máquina de estados, que direciona o seu caminho através das regras impostas nesta classe.

Tabuleiro

Esta classe representa o tabuleiro do jogo. É responsável pela verificação de uma jogada correta e de uma vitória ou empate.

IMinijogo

Esta é uma Interface que declara os métodos usados pelas classes *MinijogoMatematica* e *MinijogoDicionario*.

MinijogoDicionario

Esta classe representa o minijogo em que é pedido ao utilizador que escreva corretamente cinco palavras aleatórias num curto período de tempo. Obtém uma lista de palavras de um ficheiro *dicionario.txt* e escolhe cinco delas aleatoriamente para a realização do minijogo.

MinijogoMatematica

Nesta classe é implementado o minijogo onde o utilizador tem que acertar em resultados matemáticos aleatórios. Para esta implementação são gerados dois números de 1 a 99, que correspondem aos operandos, e outro de 1 a 4 de onde irá ser gerado o operador.

IJogador

Esta é uma Interface que declara os métodos usados nas classes *JogadorC* e *JogadorH*.

JogadorAdapter

Esta classe implementa métodos neutros, de modo a poderem ser implementados apenas os necessários em cada classe *JogadorH* e *JogadorC*.

JogadorH

Esta classe representa um jogador humano, tem nome, acesso aos minijogos, às peças especiais e créditos para a função de *undo*.

JogadorC

Nesta classe é representado um jogador artificial. O seu método de jogar é gerando um número aleatório para a coluna, não sendo utilizada qualquer tipo de inteligência artificial para cada jogada.

Relação entre classes

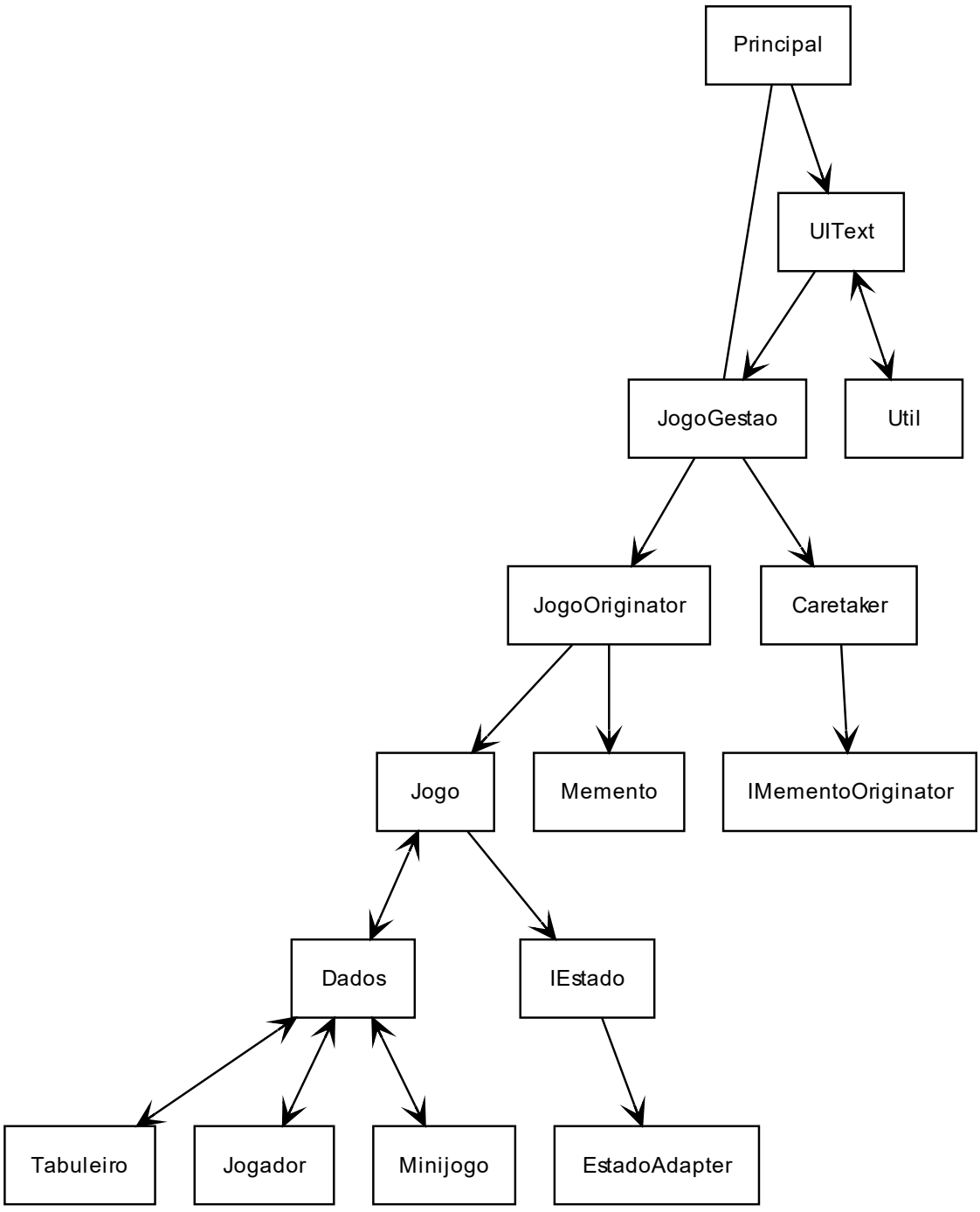

Figura 3 – Diagrama de relação entre classes

Implementação das funcionalidades

Modo Jogador vs Jogador	✔
Modo Jogador vs Computador	✔
Modo Computador vs Computador	✔
Minijogos	✔
Peças especiais	✔
Undo	✔
Gravação de jogos	✔
Carregamento de jogos com continuação	✔
Gravação de replays	✔
Carregamento de replays	✔
Log das jogadas	✔