Departamento de Engenharia Informática e de Sistemas

Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Programação Avançada 2020/2021

2ª Meta

Turma Prática Nº 2

Samuel Pires Tavares | 2019126468

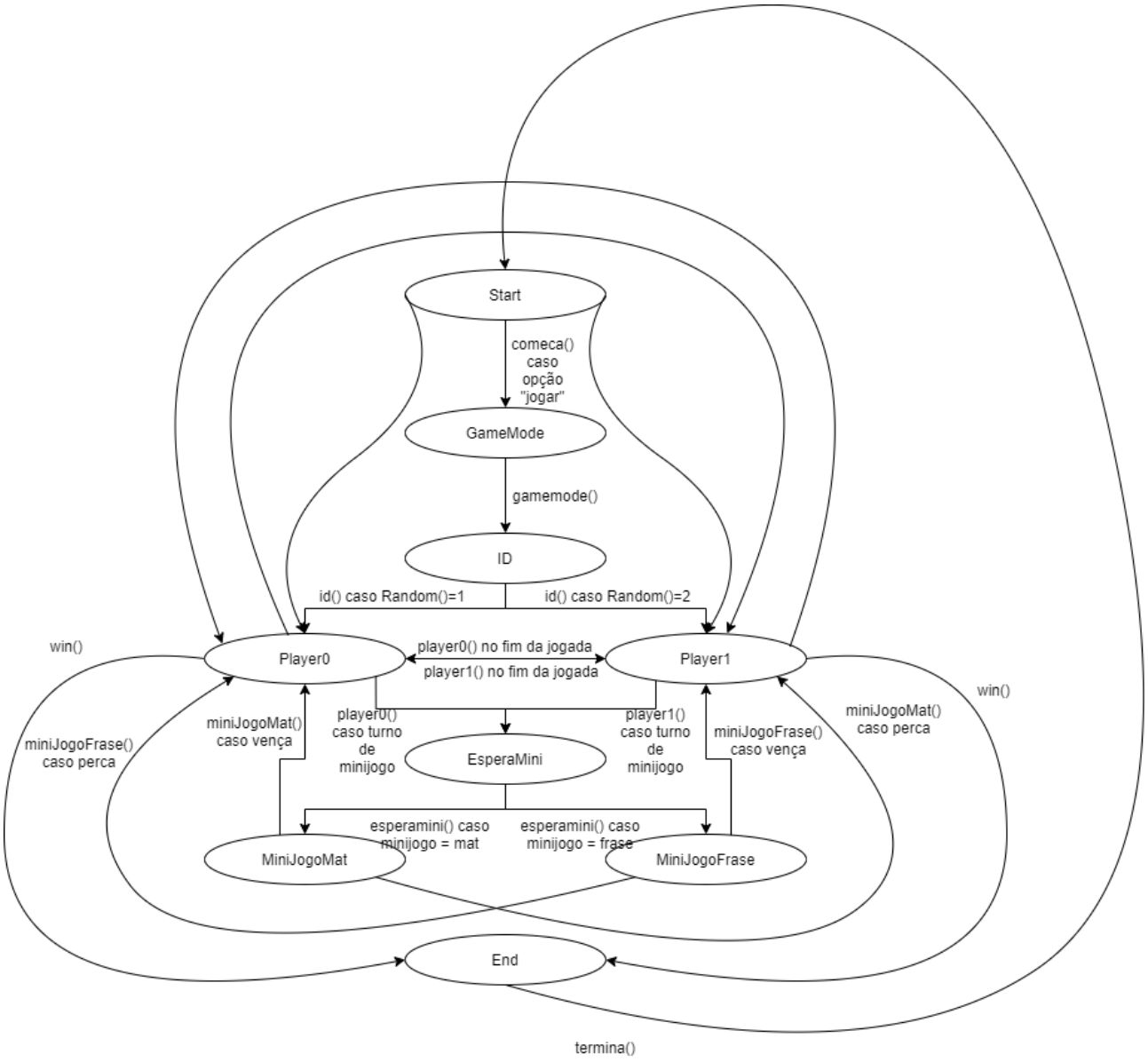
Índice

1. Opções e decisões

Este trabalho, onde era pretendida a realização de um jogo de quatro em linha com algumas funcionalidades extra, foi realizado com base em uma class Jogo onde se encontra a maior parte da lógica, e uma máquina de estados, o que é uma, se não a parte mais essencial do projeto. Para além destas classes é sempre necessário a utilização de uma interface, que nesta meta foi uma interface gráfica, presente na package GUI e dividida em vários panes, e a criação de mementos, para facilitar o gravar, carregar e replay dos ficheiros.

Os estados desenvolvidos neste projeto foram criados conforme as necessidades à medida que se ia progredindo no projeto, assim como todas as funções.

2. Diagrama da Máquina de estados

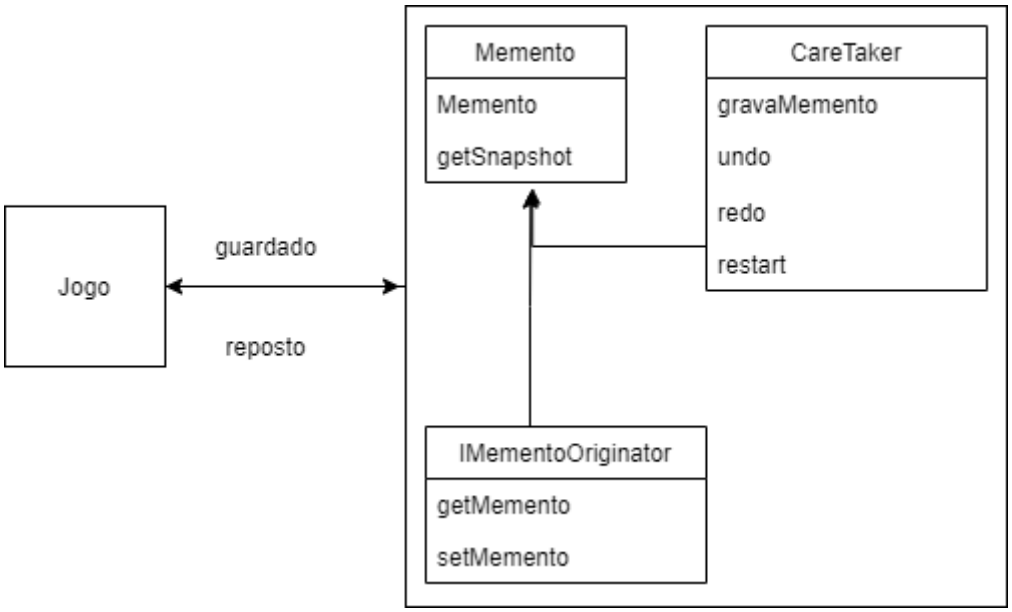


Este projeto inicia-se no estado Start, que espera uma ação do utilizador,"Jogar","Load","Replay". Caso escolha a opção "Jogar" O estado será alterado para Gamemode onde o utilizador terá a opção de escolher o modo de jovo entre PvP, PvPC ou PC v PC, independente da escolha o utilizador será reencaminhado para o estado ID onde nomeará os jogadores 1 e 2, será também efetuada uma escolha aleatória do jogador que terá o direito ao primeiro turno. O

estado passará para o estado do jogador sorteado (Player0 ou Player1), e ir-se-á alterando conforme os turnos, até que surja um minijogo, onde passa para o estado EsperaMini, onde pode aceitar ou recusar o mesmo, caso recuse, este retorna para o estado do player anterior, e caso aceite será levado para o estado do minijogo conforme as regras dadas no enunciado. Se o jogador vencer voltará ao estado do mesmo, caso perca passa a vez ao próximo jogador. Quando um jogador obtém a vitória o estado será novamente alterado, mas desta vez para o End onde o jogador escolherá se pretende voltar a jogar ou sair do jogo. Caso este decida voltar a jogar o ciclo reiniciar-se-á.

Existe também as opções de load e replay, caso no estado Start o utilizador opte por replay o joador terá a possibilidade de rever um jogo anteriormente concluído até que saia, onde será colocado no estado anteriormente guardado. Caso escolha Load o programa irá pular diretamente para o estado do Player 0 ou 1 conforme guardado no ficheiro utilizado, e irá retomar o ciclo a partir desse estado.

3. Diagrama de Outros



Como é demonstrado no diagrama, os dados do jogo são armazenados no memento, sempre que é dado um set ao memento, este guarda uma snapshot d estado do interno do Originator que pode posteriormente ser chamado com a função undo() ou redo(), dependendo se o objetivo é avançar ou retroceder.

A class CareTaker, assim como diz o nome, toma conta das operações do Memento, gerindo e guardando os mementos, servindo como uma máquina de estados particular para estes.

A Interface Originator é responsável por guardar Snapshots do seu estado nos mementos, recuperando através destes o seu estado interno.

Por sua vez, o Memento é incumbido de guardar o estado interno do Originator(snapshots), mantendo essa informação inacessível para qualquer outro objeto.

4. Classes

Jogo:

Esta é uma das classes mais importantes do programa, visto que se encontra na base do projeto. A classe Jogo possui as funções logicas, ou seja, tudo o que depender de lógica deve estar presente nesta classe.

playerList:

Esta é a classe responsável pro armazenar o array de jogadores, guardando o seu nome, número de créditos, peça especial e o seu número de identificação.

Save:

É nesta classe que se encontram as funções necessárias para o armazenamento do programa em ficheiros, tanto na função gravar, como na AutoSave que é apenas efetuada no final do jogo.

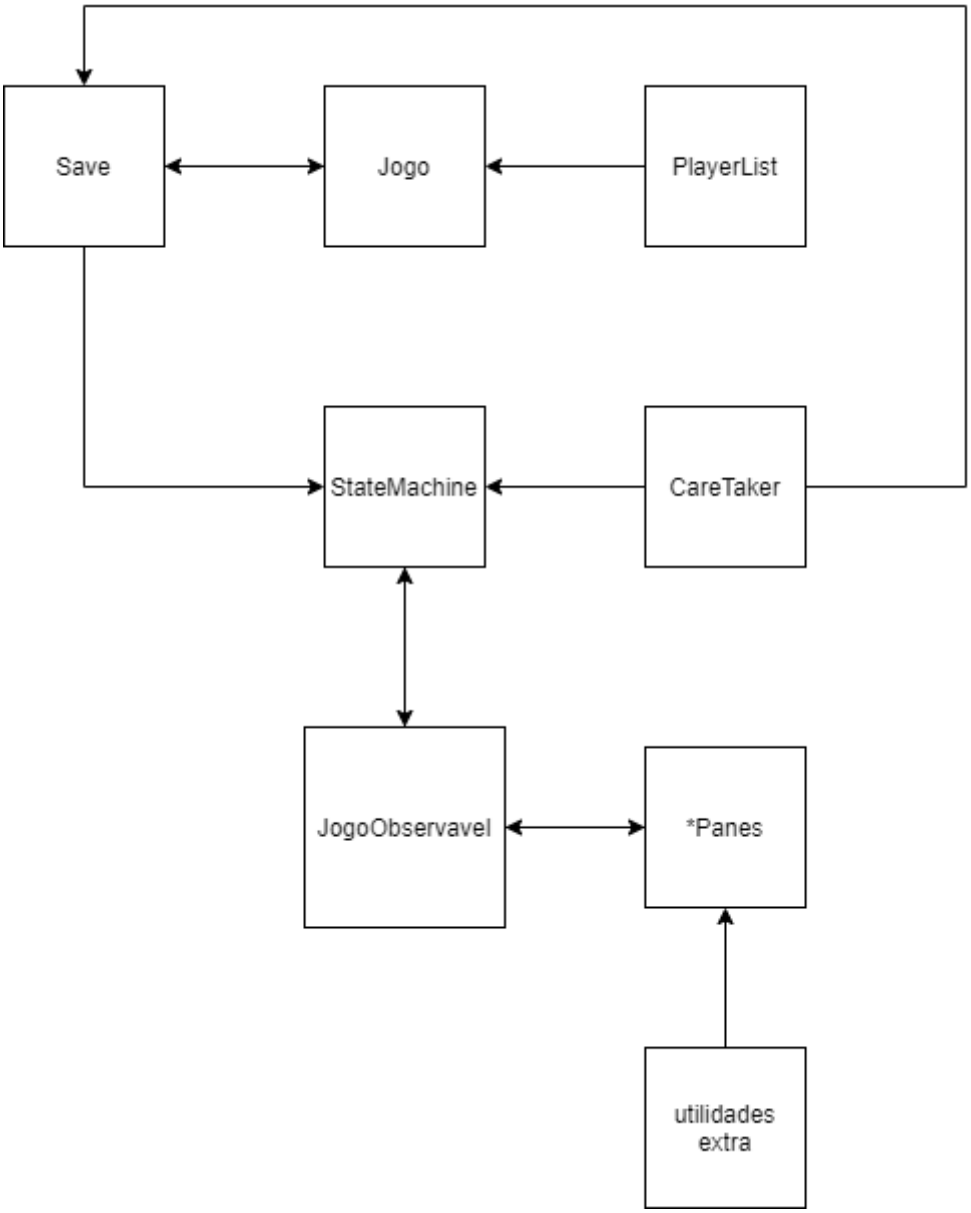
Panes

Os panes usados neste trabalho servem como uma interface gráfica, sendo que existe um para cada estado e alguns extras para realizar outras operações como mostrar o tabuleiro e certos botões.

Classes de suporte

Existem certas classes que servem para dar suporte às classes de interface, de entre elas estão a CSSManager, ImageLoader e Resources.

5. Relacionamento entre classes



- A classe Jogo recebe e guarda uma lista de jogadores fornecida pela class PlayerList.
- A class Save guarda os dados do Jogo e do CareTaker em ficheiros binários, assim como os carrega com a ajuda da StateMachine.
- As classes Jogo e CareTaker disponibilizam as suas funções à máquina de estados para que esta as comunique ao JogoObservavel.

- O JogoObservavel comunica com todos os *panes*, visto que são estes que servem de interface.
- Utilidades extra são todas aquelas que possuem funcionalidades relativas à interface, estas são utilizadas pelos *panes*.

6. Funcionalidades

Funcionalidade	Niv. Inpl.	Descrição
Máquina de estados	Implementado parcialmente	A máquina de estados ficou implementada e funcional, mas grande parte do código que devia ser encontrado nos estados está na verdade no texto, como certas validações, no entanto a maior parte da logica foi colocado no jogo e apenas chamada em outras funções.
Logs	Implementado completamente	Os Logs encontram-se funcionais e mostram todas as informações mais importantes.
Jogo	Implementado completamente	Embora os pontos anteriores se encontrem um pouco incompletos, penso que este esteja completo, visto que as funcionalidades se encontram presentes, e pelos testes efetuados aparentam estar a funcionar.
Gravar/load	Implementado completamente	Esta função aparenta funcionar corretamente.
Replay	Implementado completamente	O replay mostra todas as alterações ao tabuleiro e ao jogador na ordem correta.
PvP	Implementado completamente	Esta função funciona corretamente
PvC e CvC	Implementado completamente	As funções funcionam, e os menus são diferentes para os diferentes modos de jogo
Jogador (peça especial, créditos e nome)	Implementado completamente	Funções implementadas e funcionais
Voltar a trás	Implementado completamente	Esta funcionalidade encontrasse funcional, conforme o descrito no enunciado
Interface gráfica	Implementado completamente	A interface foi implementada e mostra-se funcional