

Departamento de Engenharia Informática e de Sistemas
Licenciatura em Engenharia Informática

Programação Avançada

Trabalho Prático 2020/2021- Fase 1



Renato Oliveira nº 2010011421

Maio de 2021

Índice

Descrição das opções e decisões tomadas na implementação	1
Diagrama Máquina de Estados	2
Descrição das classes utilizadas no programa.....	3
Jogador (Virtual ou Humano)	3
Peça	3
MiniJogo (Matematica ou Palavras).....	4
Tabuleiro	5
GerirFicheiros	5
DadosJogo	6
EstadoAdapter.....	7
AguardaInicio	7
AguardaJogada	7
AguardaFimMiniJogoMatematica	7
AguardaFimMiniJogoPortugues	7
TerminarJogo.....	7
Jogo e JogoCareTaker.....	8
Descrição do relacionamento entre classes.....	9
Funcionalidades cumpridas/Implementadas	10

Índice de Ilustrações

Figura 1 - Diagrama Máquina de Estados	2
Figura 2 - Classes Jogador, Humano e Virtual	3
Figura 3 - Classe Peca	3
Figura 4 - Classe MiniJogo, Matematica e Portugues	4
Figura 5 - Classe Tabuleiro	5
Figura 6 - Classe GerirFicheiros	5
Figura 7 - Classe DadosJogo	6
Figura 8 - Classes Máquina de Estados	7
Figura 9 - Classe Jogo e JogoCareTaker	8
Figura 10 - Diagrama de Relacionamento entre classes	9

Descrição das opções e decisões tomadas na implementação

Em relação às opções e decisões tomadas na implementação, e começando pela lógica, foram identificadas quatro classes fundamentais para os dados, uma classe memento (para gravar, carregar e fazer replay de jogos), cinco classes para os estados, uma classe para o Jogo e outra para o JogoCaretaker (ligação entre dados e ui), explicadas de forma detalhada mais à frente. Fundamentalmente irá existir um tabuleiro, que tem um conjunto de peças, dois jogadores e dois minijogos que serão geridas por uma classe DadosJogo, que por sua vez é acedida pela classe Jogo.

Em relação ao funcionamento, o jogo vai ter um menu inicial, onde o utilizador pode iniciar um jogo, fazer o replay de um jogo guardado, carregar um jogo guardado, ou terminar o jogo.

Ao iniciar o jogo, irá ser perguntado ao utilizador, se pretende iniciar o jogo com dois jogadores virtuais, um jogador virtual e outro humano, os dois jogadores humanos. Caso seja selecionada a primeira opção o jogo irá ocorrer de forma automática até ao final, numa das duas outras opções, o jogador humano vai ter de inserir um nome e um identificador para a sua peça.

Após a inserção desses dados, o jogador humano vai poder jogar numa determinada coluna, voltar atrás, usando os créditos que pretender até ao máximo de 5, usar a peça especial (caso a tenha adquirido), guardar o estado atual do jogo ou terminar o jogo.

No caso de o jogador estar na quarta jogada, pode (ou não), jogar um minijogo. Enquanto está a jogar o minijogo, não pode fazer nenhuma jogada e no final do mini jogo, pode perder a sua vez de jogar, ou então jogar a peça especial, ou jogar normalmente, guardando a peça especial para mais tarde.

Para a gravação ou carregamento do jogo, o utilizador apenas pode gravar um jogo que já tenha iniciado. Quando um jogo termina, esse jogo fica gravado com um nome predefinido para ser usado para o replay.

Para a retroceder jogadas foi usado o padrão memento, dado nas aulas práticas, e a sua função principal é ir guardando “screenshots” das jogadas de um determinado jogo em memória e depois carregá-lo conforme seja necessário.

Diagrama Máquina de Estados

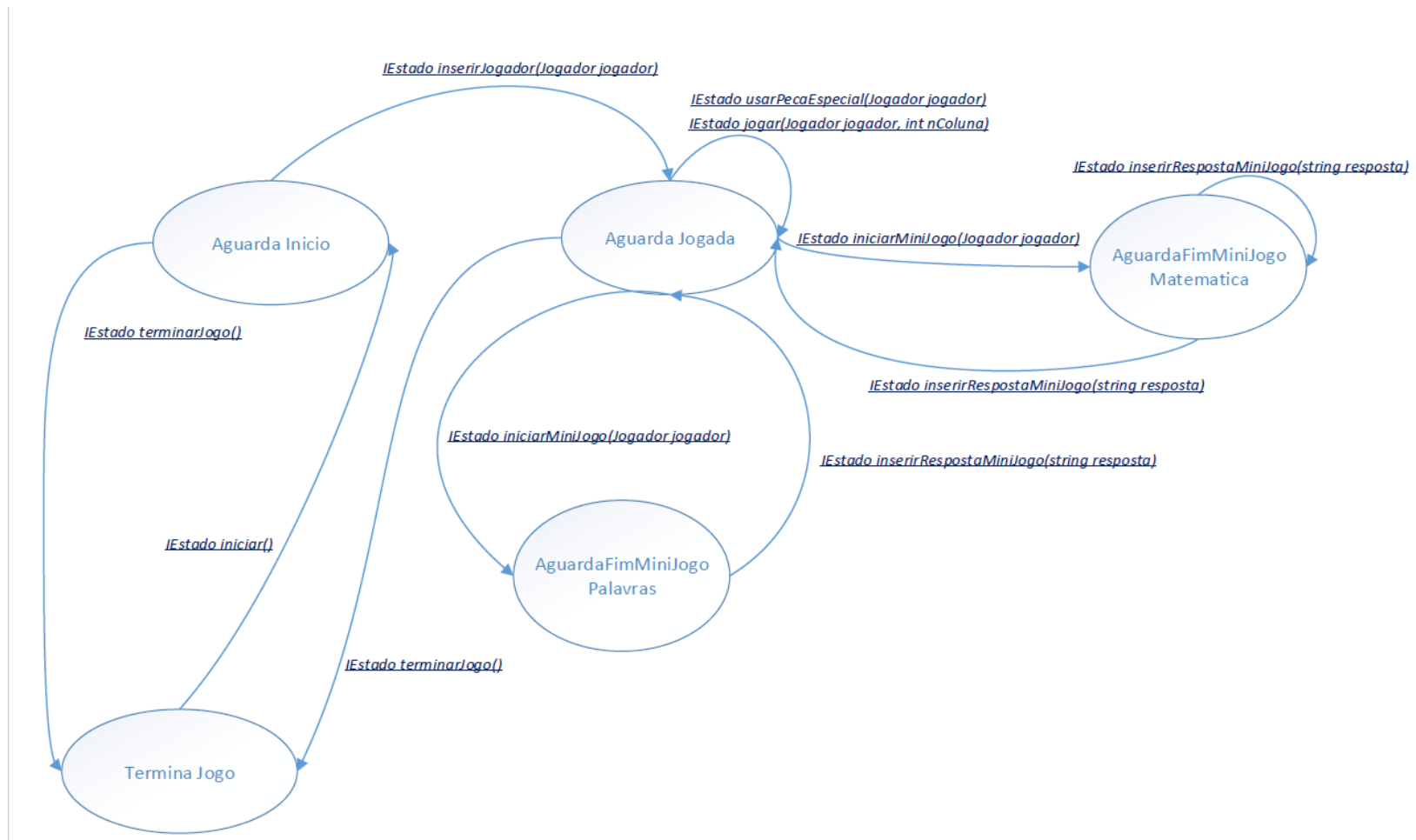


Figura 1 - Diagrama Máquina de Estados

Descrição das classes utilizadas no programa

Jogador (Virtual ou Humano)

Classe abstrata responsável por guardar informação sobre os jogadores. Cada jogador é identificado pelo seu nome, pelo seu identificador, os créditos, o número da jogada (para saber quando iniciar minijogo) e pela peça especial.

Existem duas classes “filhas” (Virtual ou Humano). No caso de o jogador ser virtual, é calculada a próxima jogada a efetuar de forma automática.

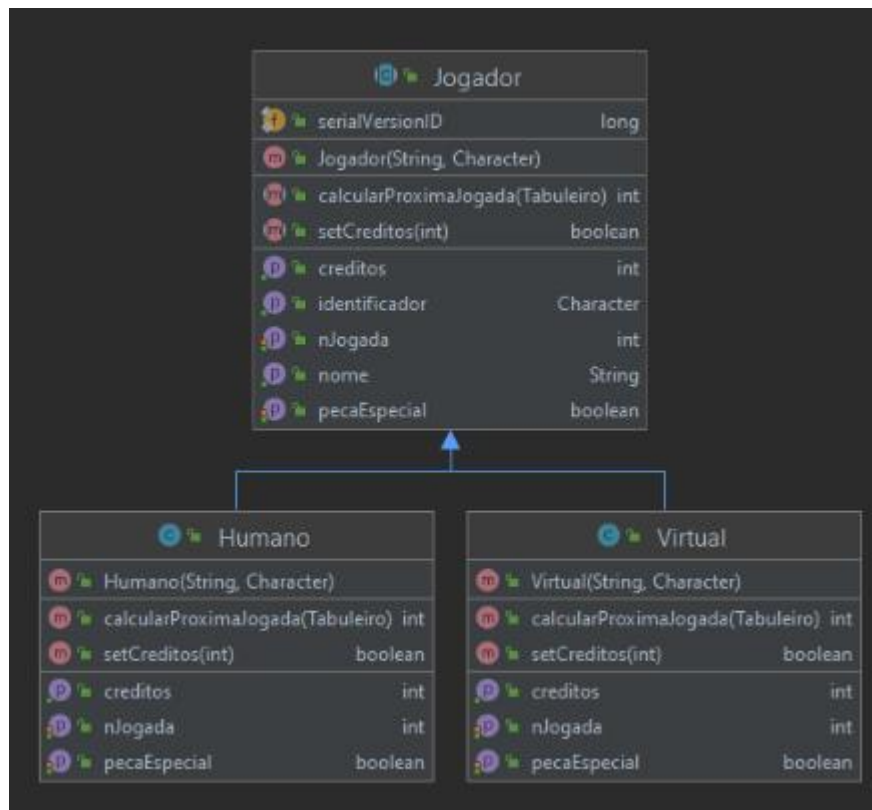


Figura 2 - Classes Jogador, Humano e Virtual

Peça

A classe peça contém a informação de uma peça. Uma peça tem uma linha, uma coluna, um identificador e uma flag (caso seja especial ou não). Para criar uma peça, deve ser passado sempre por parâmetro a linha, a coluna e o seu identificador.

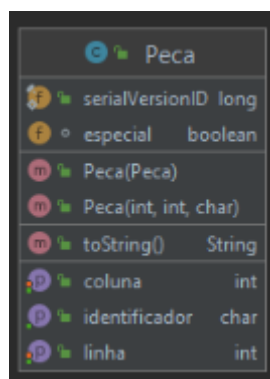


Figura 3 - Classe Peça

MiniJogo (Matematica ou Palavras)

Classe responsável pela lógica dos minijogos, guarda a informação do tempo inicial, o tempo final e o tempo passado entre o tempo inicial e o passado. É possível gerar o input do jogo, que pode ser acedido através do toString. Também é possível inserir respostas. Existem duas classes “filhas” (Portugues e Matematica), que implementam as funções de gerar o input do jogo e inserir resposta de forma diferente.

No caso da classe Portugues, vai ser carregado de um ficheiro uma listar de palavras que vão ser manipuladas de forma a gerar um input de cinco palavras e o utilizador insere uma só resposta.

No caso da classe Matematica, são gerados vários inputs durante 30 segundos e o utilizador vai inserindo resposta a resposta, até que passem os 30 segundos.

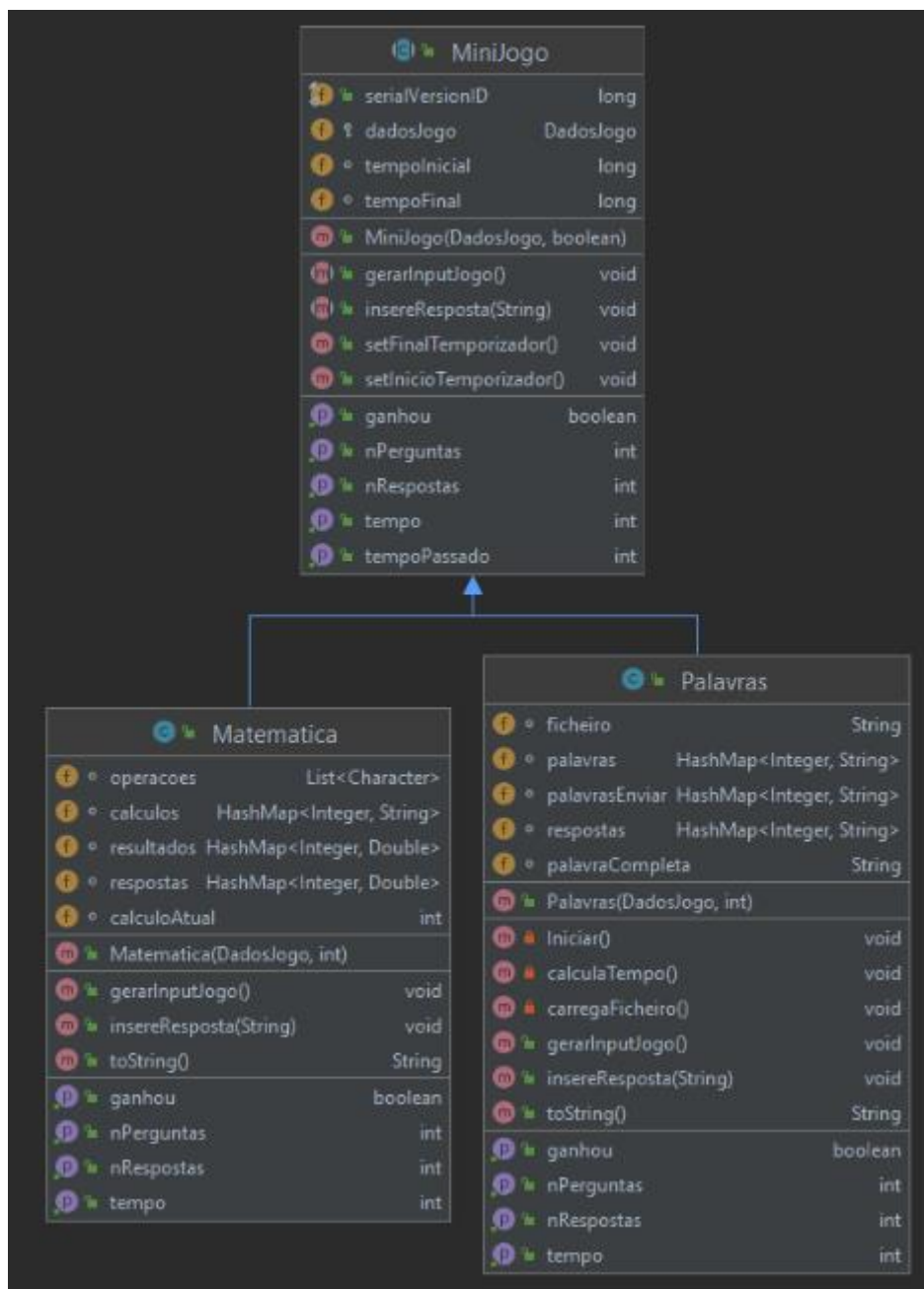
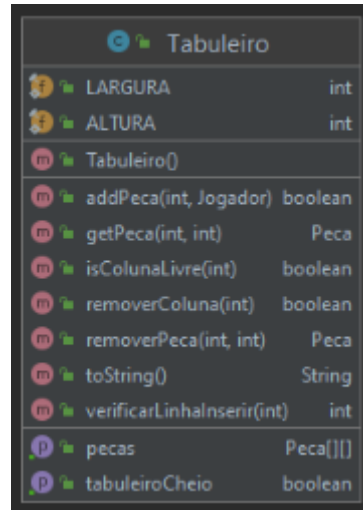


Figura 4 - Classe MiniJogo, Matematica e Portugues

Tabuleiro

A classe tabuleiro é responsável por criar o tabuleiro inicial e gerir as peças que nele existem. É possível adicionar peças, remover peças, verificar se a coluna jogada está livre, se o tabuleiro está cheio, se não está vazio, e verificar a linha a inserir (devolve o índice livre da coluna jogada).

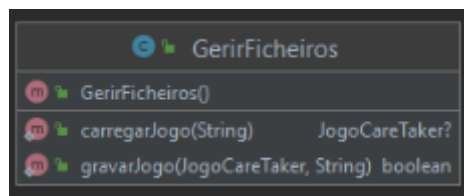


Tabuleiro		
⚙	LARGURA	int
⚙	ALTURA	int
m	Tabuleiro()	
m	addPeca(int, Jogador)	boolean
m	getPeca(int, int)	Peca
m	isColunaLivre(int)	boolean
m	removerColuna(int)	boolean
m	removerPeca(int, int)	Peca
m	toString()	String
m	verificarLinhaInserir(int)	int
p	pecas	Peca[][]
p	tabuleiroCheio	boolean

Figura 5 - Classe Tabuleiro

GerirFicheiros

Classe estática, com dois métodos estáticos, apenas para gravar e carregar um jogo.



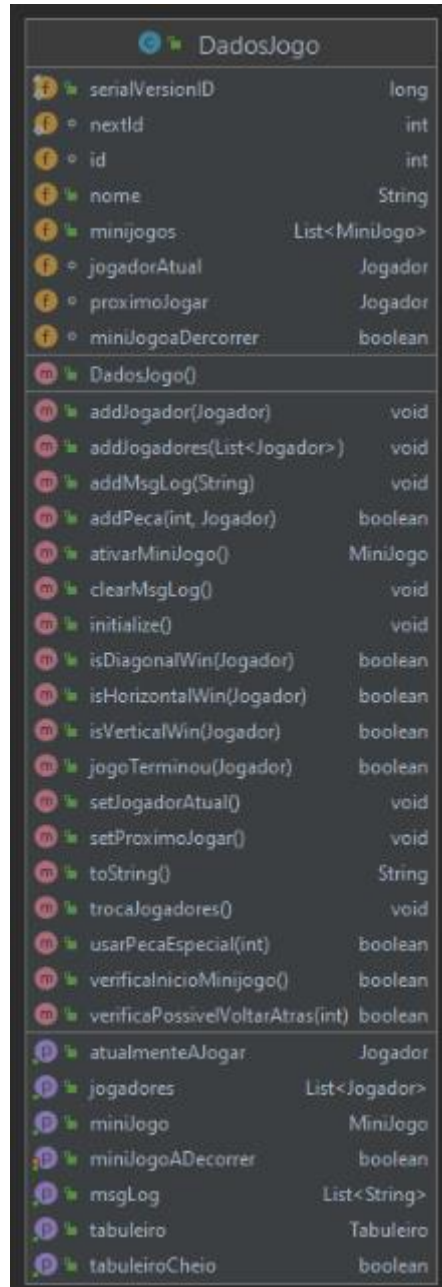
GerirFicheiros		
m	GerirFicheiros()	
m	carregarJogo(String)	JogoCareTaker?
m	gravarJogo(JogoCareTaker, String)	boolean

Figura 6 - Classe GerirFicheiros

DadosJogo

Classe responsável por manipular as classes acima referidas. É composta por um nome, (predefinido na sua criação), um tabuleiro, dois jogadores e uma lista de minijogos, e uma flag para saber se minijogo está a decorrer ou não.

Tem alguns métodos fundamentais, tais como, definir atual e próximo a jogar, adicionar um novo jogador, adicionar um peça ao tabuleiro, ativar o minijogo, verificar se é possível voltar atrás, verificar inicio de minijogo, verificar se o jogador ganhou, etc..



DadosJogo	
f	serialVersionUID long
f	nextId int
f	id int
f	nome String
f	minijogos List<MiniJogo>
f	jogadorAtual Jogador
f	proximoJogar Jogador
f	miniJogoADecorrer boolean
Methods:	
m	DadosJogo()
m	addJogador(Jogador) void
m	addJogadores(List<Jogador>) void
m	addMsgLog(String) void
m	addPeca(int, Jogador) boolean
m	ativarMiniJogo() MiniJogo
m	clearMsgLog() void
m	initialize() void
m	isDiagonalWin(Jogador) boolean
m	isHorizontalWin(Jogador) boolean
m	isVerticalWin(Jogador) boolean
m	jogoTerminou(Jogador) boolean
m	setJogadorAtual() void
m	setProximoJogar() void
m	toString() String
m	trocJogadores() void
m	usarPecaEspecial(int) boolean
m	verificaInicioMiniJogo() boolean
m	verificaPossivelVoltarAtras(int) boolean
Fields (private):	
p	atualmenteAJogar Jogador
p	jogadores List<Jogador>
p	miniJogo MiniJogo
p	miniJogoADecorrer boolean
p	msgLog List<String>
p	tabuleiro Tabuleiro
p	tabuleiroCheio boolean

Figura 7 - Classe DadosJogo

EstadoAdapter

Classe responsável por predefinir os métodos da Interface IEstado, utilizados nas classes da máquina de estados.

AguardaInicio

Classe da máquina de estados, responsável por adicionar jogadores e iniciar o jogo. O objetivo é implementar os métodos inserirJogador e terminarJogo.

AguardaJogada

Classe da máquina de estados, responsável por adicionar peças ao tabuleiro, trocar jogadores, definir jogada atual, iniciar minijogos e terminar jogo. O objetivo é implementar os métodos jogar, usarPecaEspecial e terminarJogo.

AguardaFimMiniJogoMatematica

Classe da máquina de estados, responsável por inserir as respostas do minijogo durante 30 segundos e verificar se o jogador ganhou o minijogo ou não. O objetivo é implementar o método inserirRespostaMiniJogo.

AguardaFimMiniJogoPortugues

Classe da máquina de estados, responsável por inserir a resposta do minijogo e verificar se o jogador ganhou o minijogo ou não. O objetivo é implementar o método inserirRespostaMiniJogo.

TerminarJogo

Classe da máquina de estados, responsável iniciar novo jogo. O objetivo é implementar o método iniciar, que inicia um novo jogo.

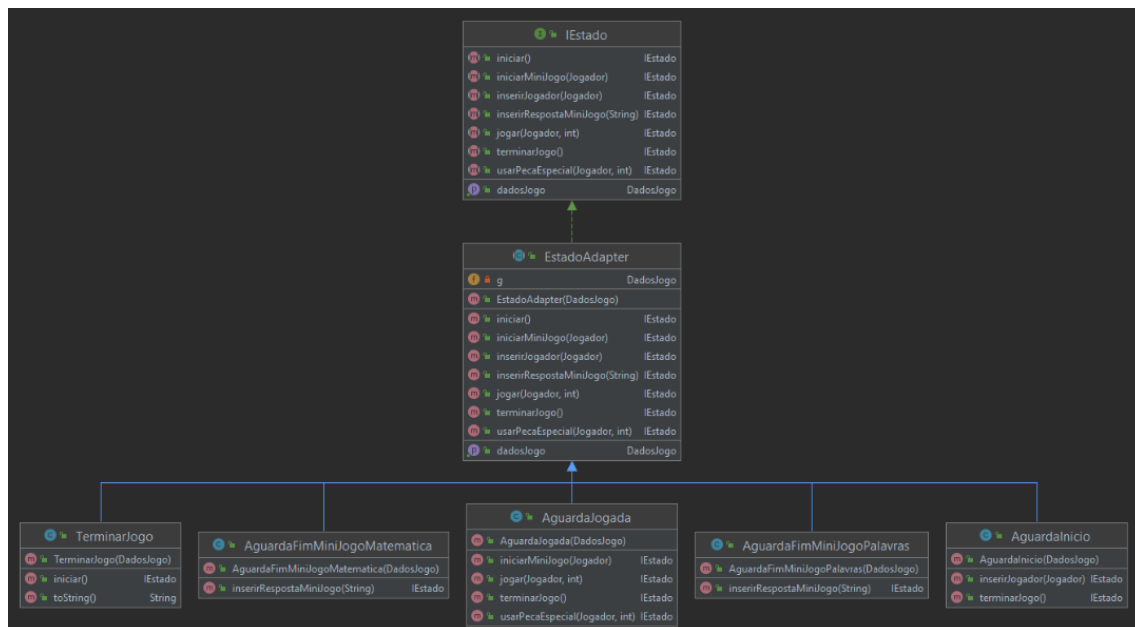


Figura 8 - Classes Máquina de Estados

Jogo e JogoCareTaker

A classe Jogo é responsável por guardar os dados de um jogo e os dados do estado desse mesmo jogo. O objetivo é a ligação dos estados com os dados do jogo, guardar e carregar mementos de um determinado estado. A classe JogoCareTaker tem a responsabilidade de guardar os dados de um jogo, um histórico de mementos para andar para trás nas jogadas e um histórico de mementos para o replay. O objetivo é a ligação da UI aos dados do jogo de forma indireta. É responsável por gerar os mementos e adicioná-los á respectiva lista, para depois essa lista ser usado no futuro. Também constrói os dados de replay para mostrar o replay de um determinado jogo.

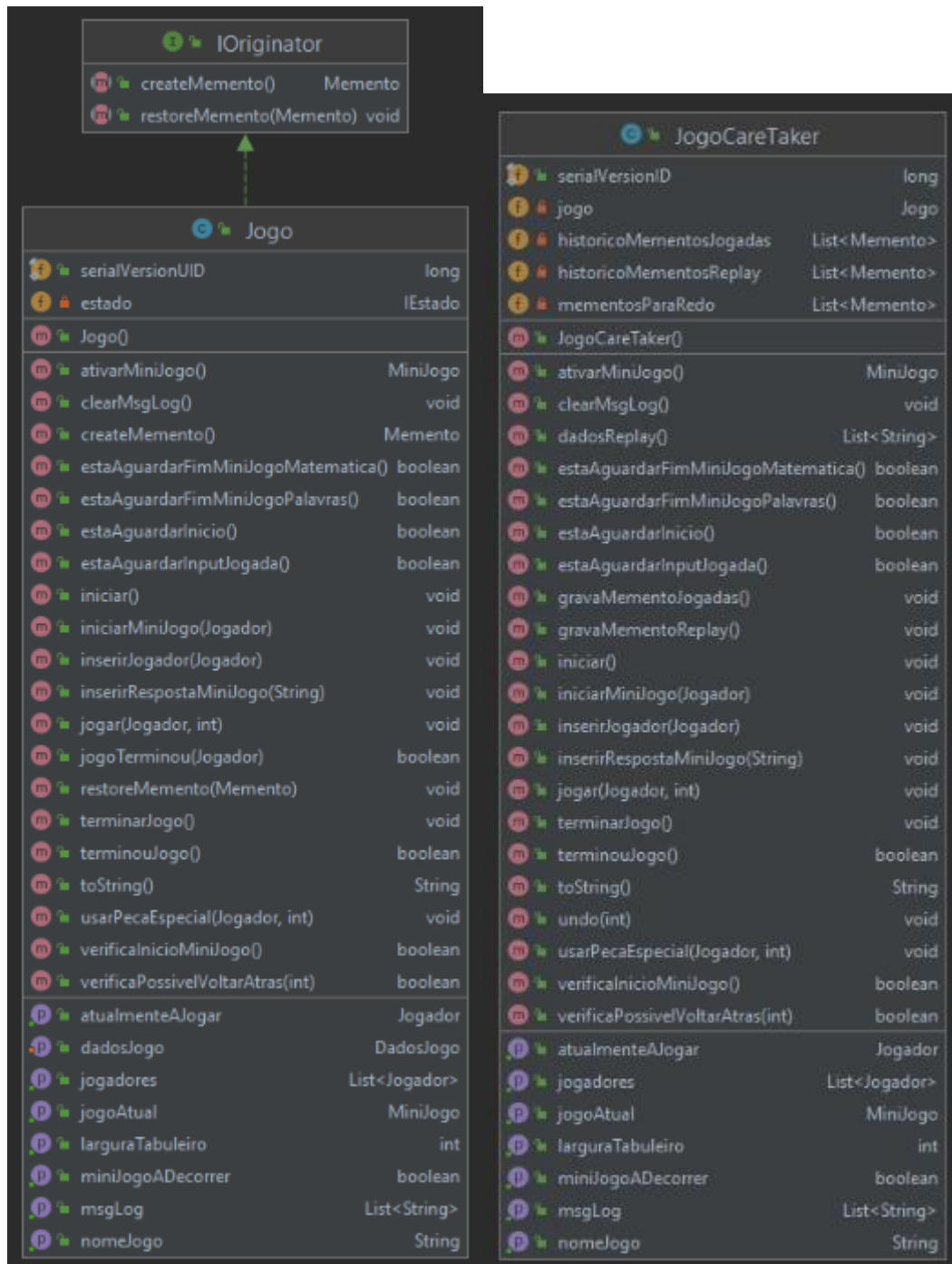


Figura 9 - Classe Jogo e JogoCareTaker

Descrição do relacionamento entre classes.

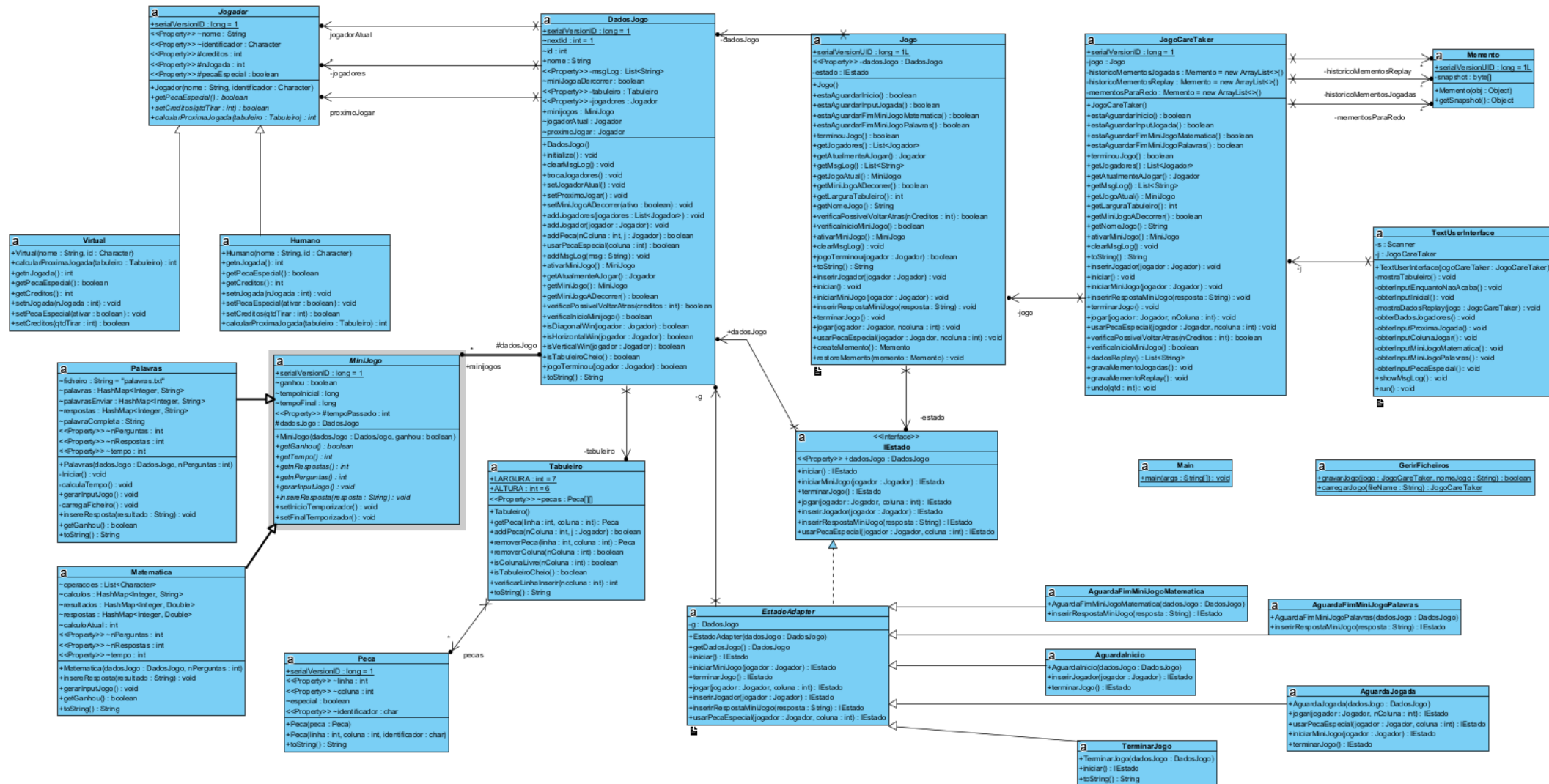


Figura 10 - Diagrama de Relacionamento entre classes

Funcionalidades cumpridas/Implementadas

Componente de trabalho	Realizado	Realizado parcialmente	Não realizado
Implementação do jogo em modo texto	✓		
Suporte para CPU vs CPU, JOG vs CPU, JOG vs JOG	✓		
Gravação/carregamento de um jogo	✓		