



Licenciatura de Engenharia Informática
Ramo de Desenvolvimento de Aplicações

Trabalho Prático de Programação Avançada



4 em Linha – 2020/2021

Daniel Carreira Pereira | a2019135953@isec.pt

1. Descrição das opções e decisões tomadas na implementação

No decorrer da implementação do jogo 4 em Linha foi necessário fazer algumas decisões na implementação.

Para o tabuleiro foi decidido que seria utilizado um ArrayList de ArrayLists de Peças. A utilização de este formato permite aceder aos diversos elementos do tabuleiro de forma relativamente simples. As Peças guardam o jogador que as jogou de forma a conseguirem ir buscar o símbolo representativo do Jogador quando é mostrado o tabuleiro.

As Peças Especiais são um número inteiro guardado no jogador.

Os minijogos são implementados na sua totalidade na sua própria classe devolvendo apenas o número de pontos. Os minijogos acontecem antes da 4ª jogada do jogador.

O AI do jogo utiliza apenas um número pseudoaleatório. Se a coluna já estiver cheia e ainda houver espaço no tabuleiro o AI gera outro número até a jogada ser válida.

Para a verificação da condição de vitória:

Na horizontal vai-se linha a linha verificar se os indexes dos jogadores que jogaram a peça nos conjuntos de 4 colunas até à Largura – 3 são iguais;

Na vertical vai-se coluna a coluna verificar se os indexes dos jogadores que jogaram a peça nos conjuntos de 4 linhas até à Altura – 3 são iguais;

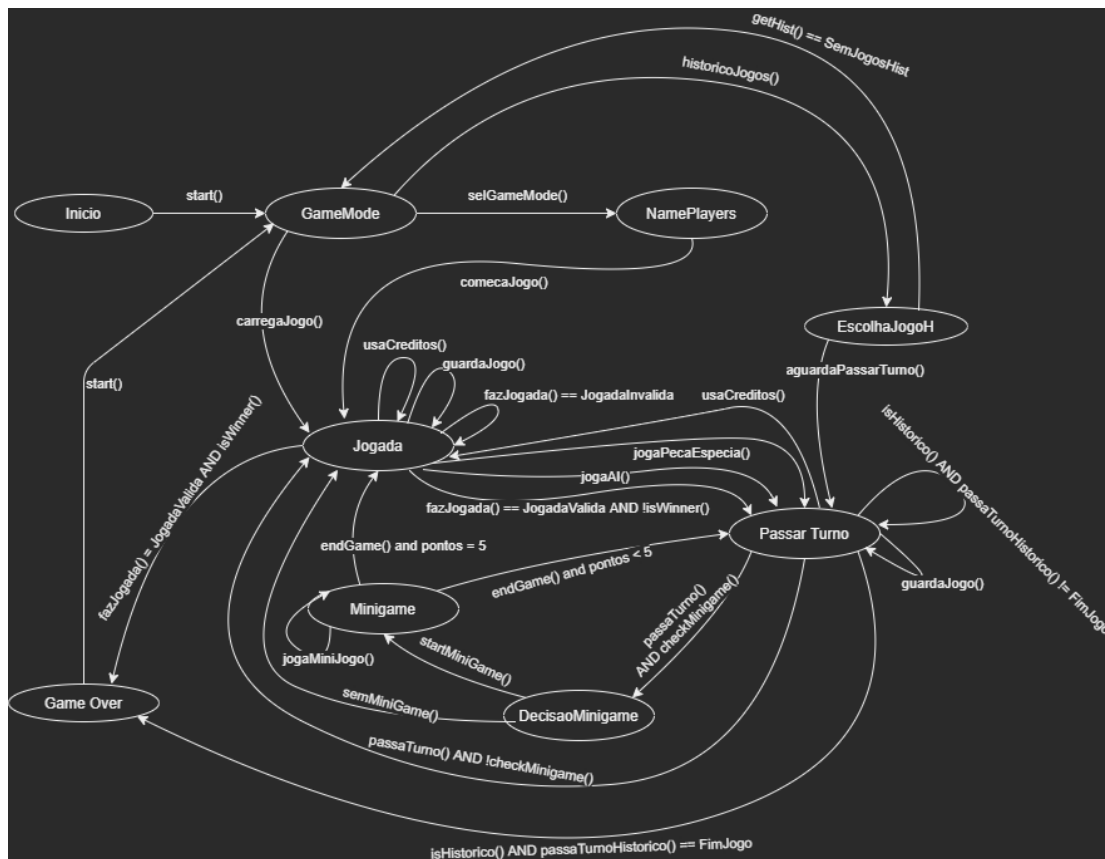
Na diagonal ascendente vai-se verificar se os indexes dos jogadores que jogaram a peça a partir da linha 3 e coluna 3 é igual ao das peças na sua diagonal à esquerda para baixo, verificando linha a linha até à Altura;

Na diagonal descendente vai-se verificar se os indexes dos jogadores que jogaram a peça a partir da linha 3 é igual ao das peças na sua diagonal à direita para baixo, verificando linha a linha até à Altura e indo apenas até à coluna Largura – 3;

Para guardar o histórico foi decidido que no final de cada jogada seria feito um clone do jogo, efetivamente guardando o estado do jogo depois da jogada ter sido feita, que seria guardado num ArrayList temporário até o jogo terminar. Quando tal acontecesse o ArrayList era adicionado ao Histórico.

Para a utilização dos Créditos será também utilizado o ArrayList temporário supracitado, carregando a jogada anterior e fazendo as alterações necessárias para o utilizador jogar. Considera-se que não é possível voltar ao estado antes de ter sido usado um crédito, por isso, não sendo possível usar um crédito na jogada imediatamente a seguir à utilização de um outro crédito. Ao utilizar o ArrayList temporário será possível no replay do jogo ver as utilizações dos créditos.

2. Diagrama da máquina de estados



O programa começa pelo estado de Início que aguarda que o utilizador leias as regras e que dê um input se pretende continuar. De seguida, através da função start() é iniciado o jogo e é dado ao utilizador um menu onde pode seleccionar se pretende começar um novo jogo e o tipo de jogo, carregar um jogo ou fazer o replay de um jogo do histórico no estado do GameMode.

- Caso o utilizador selecione um novo jogo a função selGameMode() é chamada e o utilizador será levado ao estado de NamePlayers onde terá que inserir o nome dos dois jogadores. Quando isto se verificar será chamada a função começaJogo() que levará o utilizador para o estado Jogada;
- Caso o utilizador selecione carregar um jogo será chamada a função carregaJogo() que irá carregar o jogo e direccionar o jogador para o estado Jogada;
- Caso o utilizador selecione ver um jogo do histórico será chamada a função historicoJogos() que levará o utilizador para o estado EscolhaJogoH. Neste estado serão mostrados os jogos guardados e caso não existam o utilizador é enviado de volta para o estado GameMode. Se existirem jogos o utilizador irá escolher um deles e este será carregado recorrendo à função replayHistorico() que enviará o utilizador para o estado

PassarTurno. Neste estado o utilizador poderá escolher passar o turno que está a ver através da função `passaTurnoHistorico()` que devolverá o estado `PassarTurno` até o jogo acabar, altura em que devolverá o estado `GameOver`;

No estado `Jogada` o utilizador poderá escolher de entre várias opções:

- Usar `Creditos` que invocará a função `usaCreditos()` que levará o utilizador de volta para o estado `Jogada` depois de carregar um momento anterior do jogo.
- Guardar Jogo que invocará a função `guardaJogo()` que irá guardar o momento atual do jogo num ficheiro e retornará o utilizador para o estado `Jogada`.
- Fazer `Jogada` que invocará a função `fazJogada()` / `jogaAI()`, dependendo se o jogador é humano ou AI, que irá devolver o estado `Jogada` se a jogada for inválida, ou irá devolver o estado `PassarTurno` se a jogada for válida e a função `isWinner()` devolva falso. Caso a função `isWinner()` devolva verdade o utilizador será levado para o estado `GameOver`.
- Jogar `Peca Especial` que invocará a função `jogaPecaEspecial()` que devolverá o estado `PassarTurno` depois de jogar a peça especial.

No estado `PassarTurno` o utilizador poderá escolher entre as opções:

- Passar Turno que invocará a função `passaTurno()` que irá verificar se o jogador seguinte poderá jogar o minijogo.
 - Caso o utilizador possa jogar o minijogo será levado para o estado `DecisaoMinigame` onde poderá escolher jogar o minijogo. Caso não o faça será levado através da função `semMinigame()` para o estado `Jogada`. Caso pretenda jogar será levado para o estado `Minigame` pela função `jogaMinijogo()`. Se o utilizador obter 5 pontos será levado para o estado `Jogada` através da função `continuaJogada()`. Se não conseguir obter 5 pontos irá perder a sua vez de jogar e será levado para o estado `PassarTurno` pela função `aguardaPassarTurno()`.
 - Caso o utilizado não esteja em condições de jogar o minijogo será levado pela função `passaTurno()` para o estado `Jogada`.
- Guardar Jogo que invocará a função `guardaJogo()` que irá guardar o momento atual do jogo num ficheiro e retornará o utilizador para o estado `Passar Turno`.
- Usar `Creditos` que invocará a função `usaCreditos()` que levará o utilizador para o estado `Jogada` depois de carregar um momento anterior do jogo.

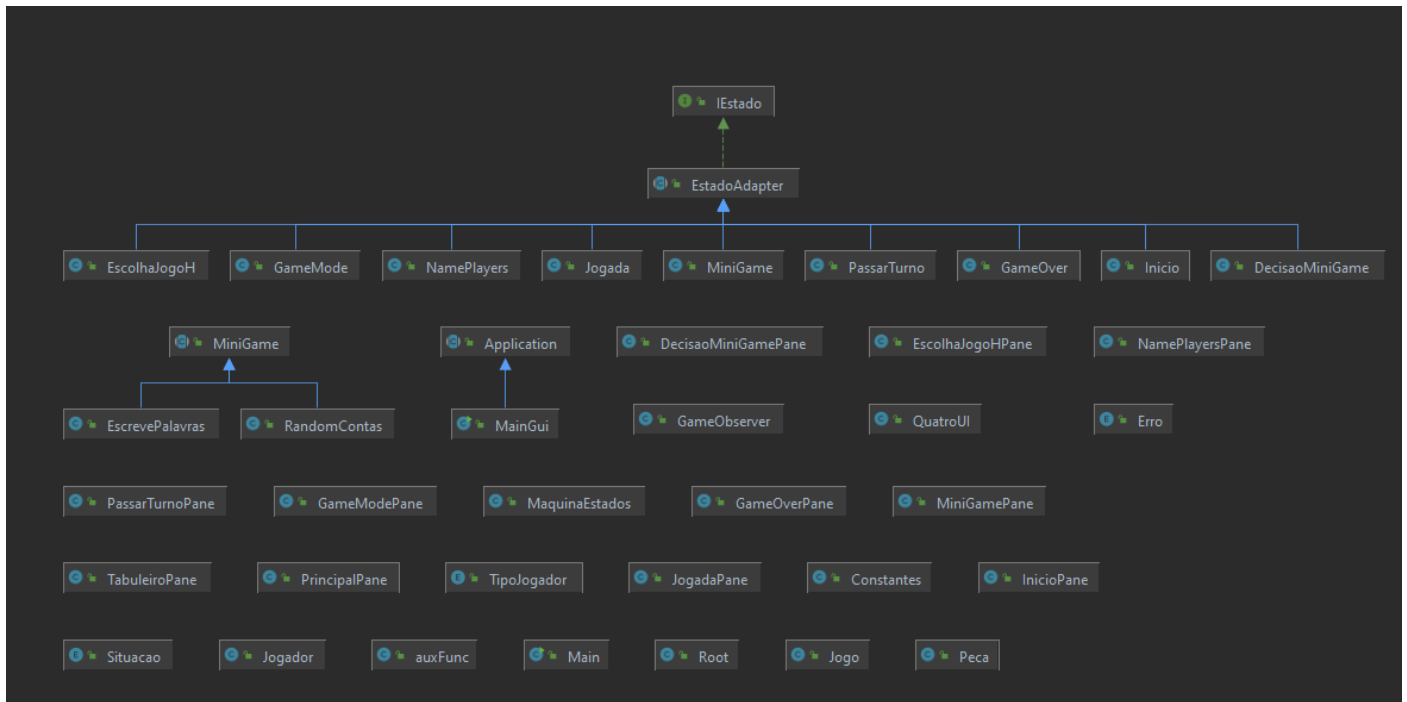
No estado `GameOver` é dada a opção ao utilizador de continuar a utilizar o programa ou de terminar a sua execução. Caso o utilizador continuar a utilizar o programa será chamada a função `start()` que levará o utilizador de volta ao estado `GameMode`.

3. A descrição das classes utilizadas no programa

- Classe auxFunc -> Funções auxiliares para interação com o utilizador
- Classe QuatroUI -> Classe que realiza a comunicação com o utilizador
- Classe Minigame -> Classe abstrata de onde se vão estender os vários minijogos
- Classe EscrevePalavras -> Classe que implementa o minijogo das palavras
- Classe RandomContas -> Classe que implementa o minijogo das contas
- Classe Jogador -> Classe que guarda e altera dados relativos ao jogador
- Classe Peca -> Classe que representa as peças no tabuleiro
- Classe Jogo -> Classe que lida com o processamento do jogo
- Classe MaquinaEstados -> Classe que liga o UI ao Jogo
- Classe DecisaoMiniGame -> Classe que representa o estado enquanto o utilizador decide se pretende jogar o minijogo
- Classe EscolhaJogoH -> Classe que representa o estado enquanto o utilizador escolhe um dos jogos do histórico para dar replay
- Classe GameMode -> Classe que representa o estado enquanto o utilizador decide se quer começar um novo jogo, carregar um jogo ou ver um jogo do histórico
- Classe GameOver -> Classe que representa o estado enquanto o utilizador decide se quer permanecer na aplicação ou terminá-la
- Classe Inicio -> Classe inicial que aguarda que o utilizador leia as regras e decide se quer continuar a usar a aplicação
- Classe Jogada -> Classe que representa o estado quando é dada a opção de o utilizador jogar uma peça, guardar o jogo, jogar créditos ou peças especiais
- Classe MiniGame -> Classe que representa o estado enquanto o utilizador joga o minijogo
- Classe NamePlayers -> Classe que representa o estado enquanto o utilizador nomeia os jogadores de um novo jogo
- Classe PassarTurno -> Classe que representa o estado enquanto o utilizador não passa o turno
- Classe EstadoAdapter -> Classe que implementa a interface IEstado e da qual derivam as classes dos estados
- Classe MainGUI -> Classe onde é iniciada a execução do programa com o JavaFX
- Classe Main -> Classe onde é iniciada a execução do programa para o UI de texto
- Classe PrincipalPane -> Classe que representa o GUI da aplicação
- Classe TabuleiroPane -> Classe responsável por mostrar o tabuleiro na aplicação
- Classe Constantes -> Classe com algumas constantes úteis no decorrer do programa
- Classe Root -> Classe que cria a janela principal
- Classe InicioPane -> Classe que representa o GUI de parte da aplicação no estado Inicio

- Classe GameModePane -> Classe que representa o GUI de parte da aplicação no estado GameMode
- Classe NamePlayersPane -> Classe que representa o GUI de parte da aplicação no estado NamePlayers
- Classe JogadaPane -> Classe que representa o GUI de parte da aplicação no estado Jogada
- Classe PassarTurnoPane -> Classe que representa o GUI de parte da aplicação no estado PassarTurno
- Classe GameOverPane -> Classe que representa o GUI de parte da aplicação no estado GameOver
- Classe DecisaoMiniGamePane -> Classe que representa o GUI de parte da aplicação no estado DecisaoMiniGame
- Classe MiniGamePane -> Classe que representa o GUI de parte da aplicação no estado MiniGame
- Classe EscolhaJogoHPane -> Classe que representa o GUI de parte da aplicação no estado EscolhaJogoH
- Classe GameObserver -> Classe que representa uma camada intermédia entre o GUI e a máquina de estados

4. Descrição do relacionamento entre as classes



5. Funcionalidades implementadas

Funcionalidades	Implementado	Por Implementar
Implementação do jogo em modo de texto	X	
Suporte para dois jogadores humanos	X	
Suporte para dois jogadores AI	X	
Suporte para um jogador humano e um jogador AI	X	
Gravação do jogo	X	
Carregamento do jogo	X	
Peças Especiais	X	
Minijogo RandomContas	X	
Minijogo EscrevePalavras	X	
Créditos	X	
Histórico	X	
Logs Máquina Estados	X	
Implementação do jogo em modo gráfico	X	